Aula 00 (Profs. Paolla Ramos e Raphael Lacerda)

TJ-MS (Técnico de Nível Superior - Analista de Sistemas Computacionais - Analista de Suporte de TI) Desenvolvimento de Software - 2024

Autor:
Paolla Ramos
(Pós-Edital)

08 de Fevereiro de 2024

Índice

1) DevOps - Teoria 3
2) DevOps - Questões Comentadas 48
3) DevOps - Lista de Questões 77
4) Docker - Teoria 89
5) Docker - Questões Comentadas 130
6) Docker - Lista de Questões 151
7) Kubernetes - Teoria 161
8) Kubernetes - Questões Comentadas 214
9) Kubernetes - Lista de Questões 238
10) OpenShift - 100% - Teoria 249
11) OpenShift - 100% - Questões Comentadas 251
12) OpenShift - 100% - Lista de Questões 253
13) Rancher - Teoria 255
14) Rancher - Questões Comentadas 266
15) Rancher - Lista de Questões 268

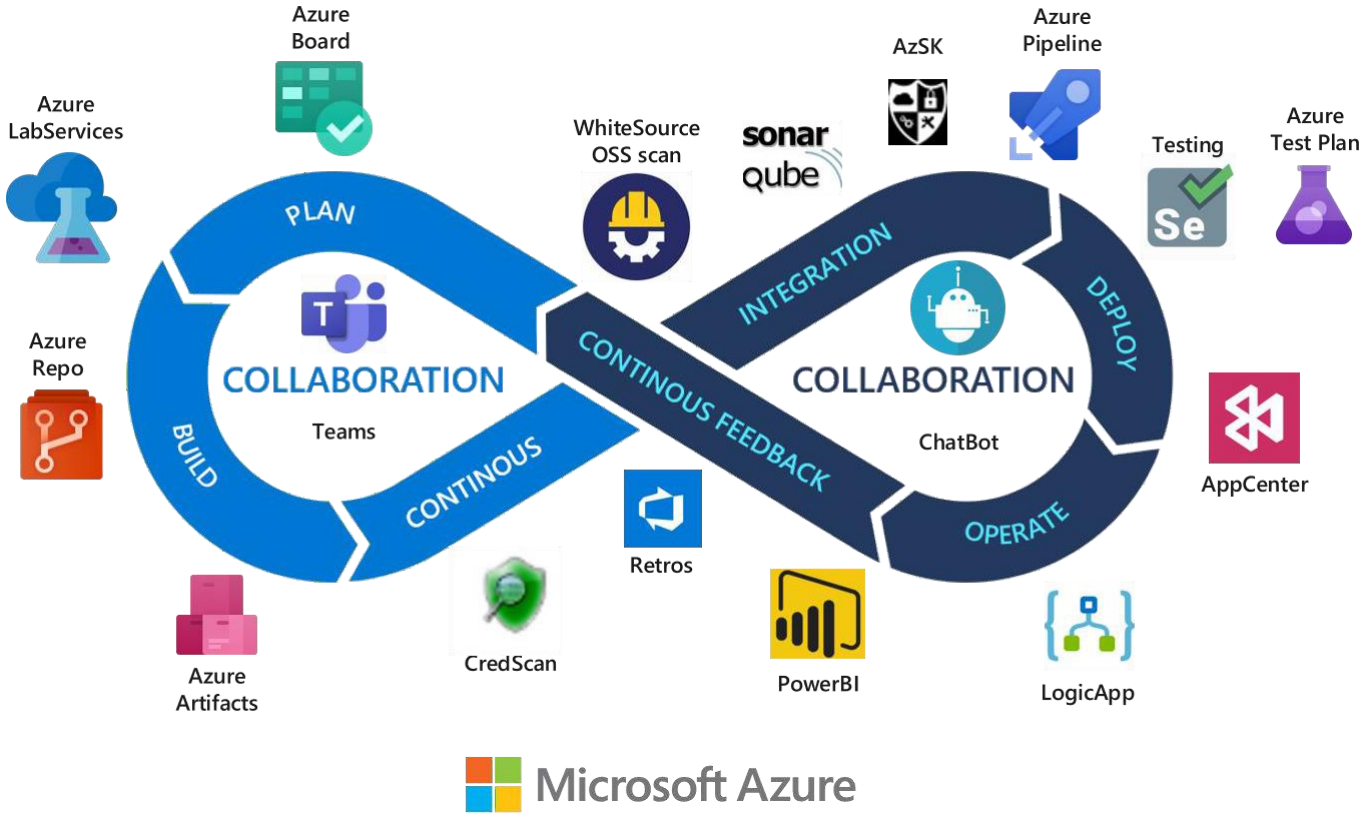
Sumário

Apresentação da Aula 2
DevOps 4
Conceitos Básicos 4
Princípios DevOps 6
Ciclo de Vida 8
Deployment Contínuo 10
Entrega Contínua 12
Ferramentas de DevOps 18
Colaboração 19
Gerenciamento de Código Fonte 20
Automação de Banco de Dados 21
Integração Contínua 22
Build (Construção) 28
Implantação 29
Operações 30
Ferramentas de registro (logging). 31
Ferramentas de monitoramento. 31
DevOps no Mundo Real 32
Testes 33
Gerenciamento de Configuração 35
Implantação 36
Containers 37
Orquestração de Lançamento 39
Cloud 40
AIOps 41
Analytics 41
Monitoramento 42
Segurança 43
Colaboração multidisciplinar 43
Referências 45

APRESENTAÇÃO DA AULA

Olá, pessoal! Hoje vamos falar sobre DevOps e como essa abordagem tem transformado a forma como desenvolvemos e entregam software. DevOps é uma cultura e conjunto de práticas que integra os times de desenvolvimento (Dev) e operações (Ops), buscando maior colaboração, automação e eficiência em todo o ciclo de vida do software.

Com o DevOps, a ideia é que os desenvolvedores e as equipes de operações trabalhem de forma conjunta desde o início do projeto até a sua implantação e manutenção. Isso significa que a colaboração é incentivada em todas as etapas do processo, permitindo uma comunicação mais eficiente e uma entrega mais rápida e confiável do software.



Uma das principais características do DevOps é a automação. Através de ferramentas e práticas automatizadas, é possível acelerar o desenvolvimento, testes, implantação e monitoramento do software. Por exemplo, é comum utilizar ferramentas de integração contínua (CI) e implantação contínua (CD) para automatizar os processos de construção, testes e implantação do software em diferentes ambientes.

Além disso, o DevOps também promove a cultura de feedback contínuo e melhoria. Isso significa que os times estão sempre buscando formas de aprender com as experiências, medindo e

monitorando o desempenho do software em produção e realizando ajustes e melhorias constantes.

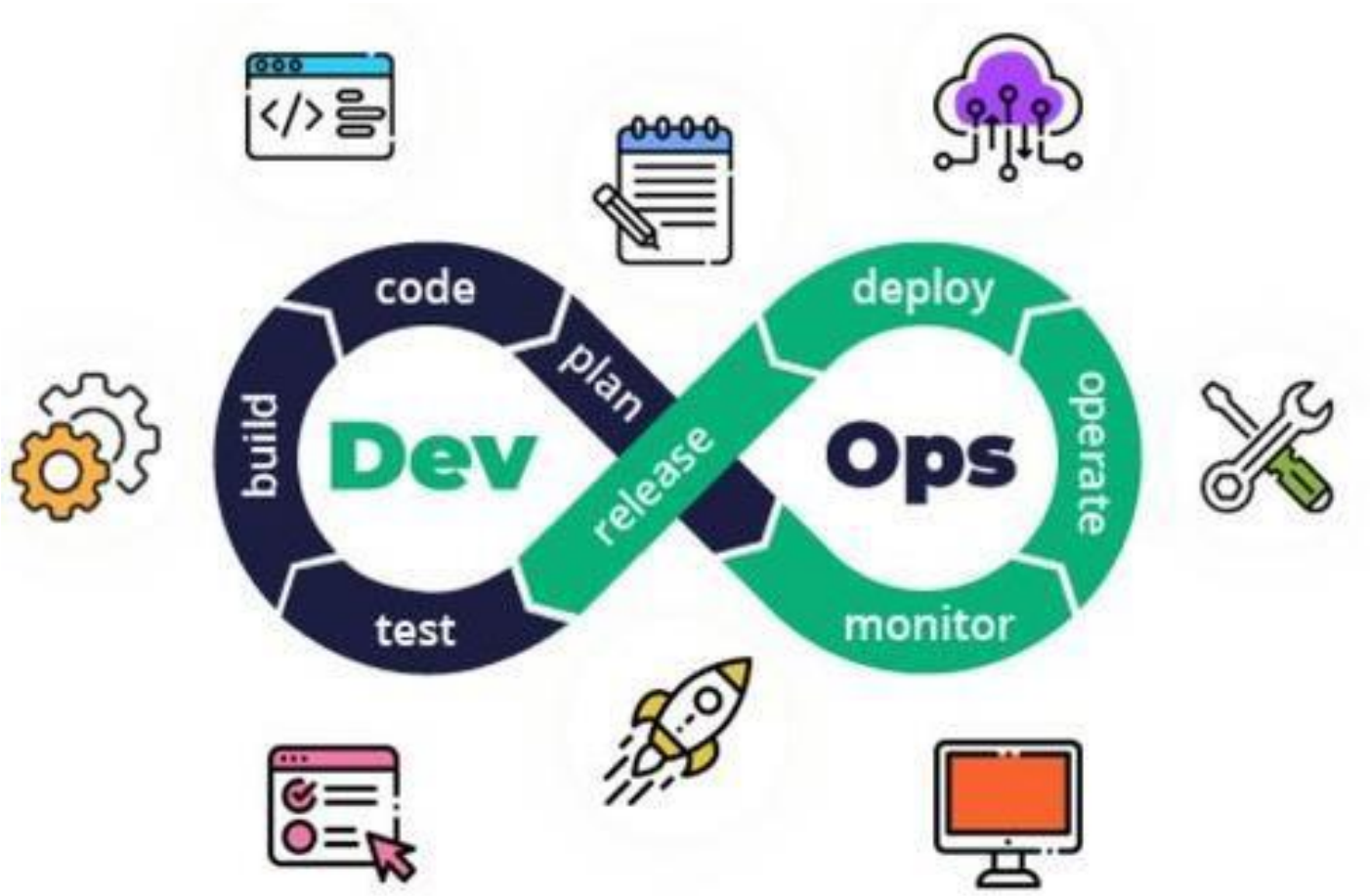
Uma das principais vantagens do DevOps é a capacidade de entregar software de forma mais rápida e frequente. Através da automação e colaboração, é possível reduzir o tempo entre o desenvolvimento de uma funcionalidade e sua disponibilização para os usuários. Isso permite que as empresas sejam mais ágeis, respondendo rapidamente às demandas do mercado e obtendo feedback mais cedo.

Outro benefício importante do DevOps é a maior estabilidade e confiabilidade do software. Ao integrar as equipes de desenvolvimento e operações, é possível antecipar e resolver problemas de forma mais eficiente, garantindo que o software seja implantado de maneira segura e confiável. Além disso, a automação dos processos reduz a chance de erros humanos e ajuda a manter a consistência entre os ambientes de desenvolvimento, teste e produção.

No entanto, implementar o DevOps não é apenas uma questão de adotar ferramentas e práticas específicas. É necessário promover uma mudança cultural dentro das equipes, incentivando a colaboração, a comunicação e a busca constante pela melhoria. Também é importante investir em treinamento e capacitação para que os membros das equipes possam adquirir as habilidades necessárias para trabalhar nesse novo ambiente colaborativo. Agora é hora de debruçar nos estudos e aprender sobre DevOps!

DevOps

Conceitos Básicos

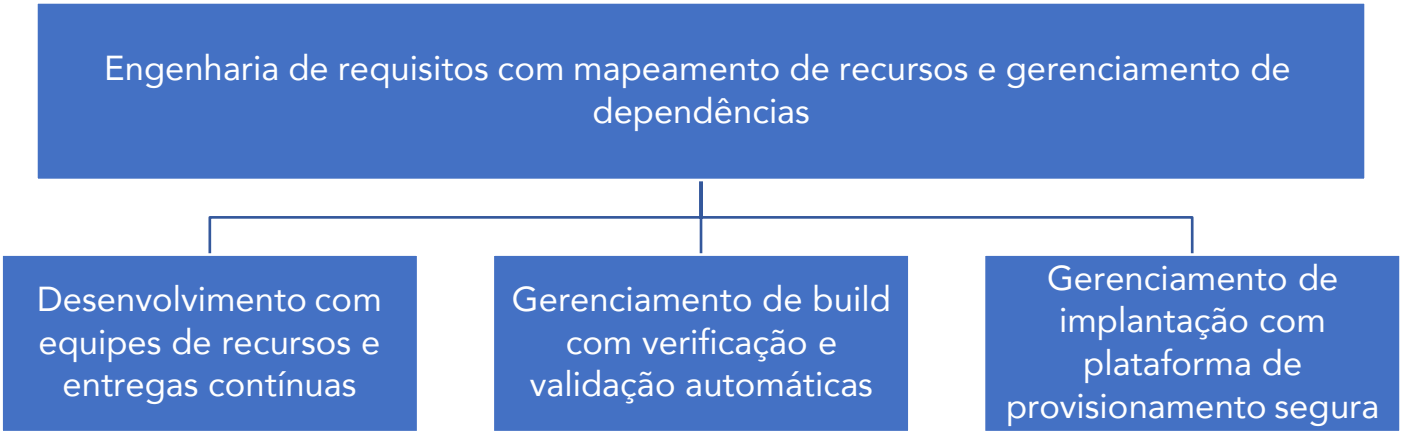


De acordo com Christof Ebert, DevOps é sobre desenvolvimento rápido e flexível e provisão de processos de negócios. Ele integra de forma eficiente o desenvolvimento, entrega e operações, facilitando uma conexão ágil e fluida desses silos tradicionalmente separados.

DevOps é uma abordagem que integra desenvolvimento e operações para tornar o processo de entrega de software mais eficiente. Em vez de ter equipes separadas trabalhando em suas próprias áreas, o DevOps promove a colaboração entre desenvolvedores, profissionais de qualidade e operações. Assim, DevOps integra os dois mundos do desenvolvimento e das operações, utilizando o desenvolvimento automatizado, implantação e monitoramento da infraestrutura. É uma mudança organizacional em que, em vez de grupos distribuídos em silos executando funções separadamente, equipes multidisciplinares trabalham em entregas contínuas de recursos operacionais. Essa abordagem ajuda a entregar valor de forma mais rápida e contínua, reduzindo

problemas decorrentes da falta de comunicação entre os membros da equipe e acelerando a resolução de problemas.

DevOps é uma mudança cultural em que as equipes adotam uma cultura de engenharia de software, fluxo de trabalho e conjunto de ferramentas que elevam os requisitos operacionais ao mesmo nível de importância que a arquitetura, design e desenvolvimento. Os desenvolvedores que criam e executam o software têm uma maior compreensão dos requisitos e necessidades dos usuários. Os valores de uma cultura de DevOps incluem o aumento da transparência, a comunicação e a colaboração entre equipes. O esquema 1 mostra o processo genérico, que tem como objetivo integrar melhor os processos de negócios de desenvolvimento, produção e operações com a tecnologia adequada.



Em vez de permanecer com conceitos de processos altamente artificiais que nunca fluirão, as organizações estabelecem uma entrega contínua com pequenas atualizações. Empresas como Amazon e Google lideraram essa abordagem, alcançando tempos de ciclo de minutos. Obviamente, o tempo de ciclo alcançável depende das restrições ambientais e do modelo de implantação; um único serviço em nuvem é mais fácil de facilitar do que entregas reais de produtos reais.

DevOps pode ser aplicado a modelos de entrega muito diferentes, mas deve ser adaptado ao ambiente e à arquitetura do produto. Nem todos os produtos facilitam a entrega contínua, por exemplo, em sistemas críticos de segurança. No entanto, mesmo em ambientes restritos como esse, as atualizações podem ser planejadas e entregues rapidamente e de forma confiável, como mostra a evolução recente do software automotivo over-the-air. Além da entrega baseada em nuvem altamente segura, tais modelos de entrega exigem mudanças de arquitetura e hardware dedicados. Um exemplo é um

controlador hot-swap em que uma metade está operacional e a outra metade constrói as próximas atualizações, que são trocadas para o modo ativo após procedimentos de segurança e verificação aprofundados. O DevOps para sistemas embarcados é mais desafiador do que para nuvem e serviços de TI, pois tenta combinar código e arquitetura legados com entrega contínua.

Princípios DevOps

Os princípios do DevOps, conforme enunciados por Jez Humble e David Harley, estão alinhados com a ideia de uma entrega de software mais eficiente e colaborativa. Vamos revisar cada um deles:

- Crie um processo repetível e confiável para entrega de software: Este princípio enfatiza a importância de ter um processo consistente e confiável para a entrega de software. A ideia é que a entrega não seja um evento traumático, mas sim algo rotineiro e previsível. Isso envolve ter etapas automatizadas, garantindo que a entrega seja tão simples quanto pressionar um botão.

- Automatize tudo que for possível: Esse princípio está intrinsecamente ligado ao primeiro. Ele defende que todas as etapas do processo de entrega de software devem ser automatizadas, desde a compilação do código até a configuração dos servidores e a implantação do sistema. A automação permite uma entrega mais rápida, consistente e livre de erros humanos.

- Mantenha tudo em um sistema de controle de versões: Esse princípio enfatiza a importância de manter todo o código fonte, arquivos de configuração, scripts, documentação e outros artefatos relacionados ao software em um sistema de controle de versões, como o Git. Isso facilita o rastreamento de alterações, o trabalho colaborativo e a possibilidade de retornara versões anteriores do sistema, caso necessário.

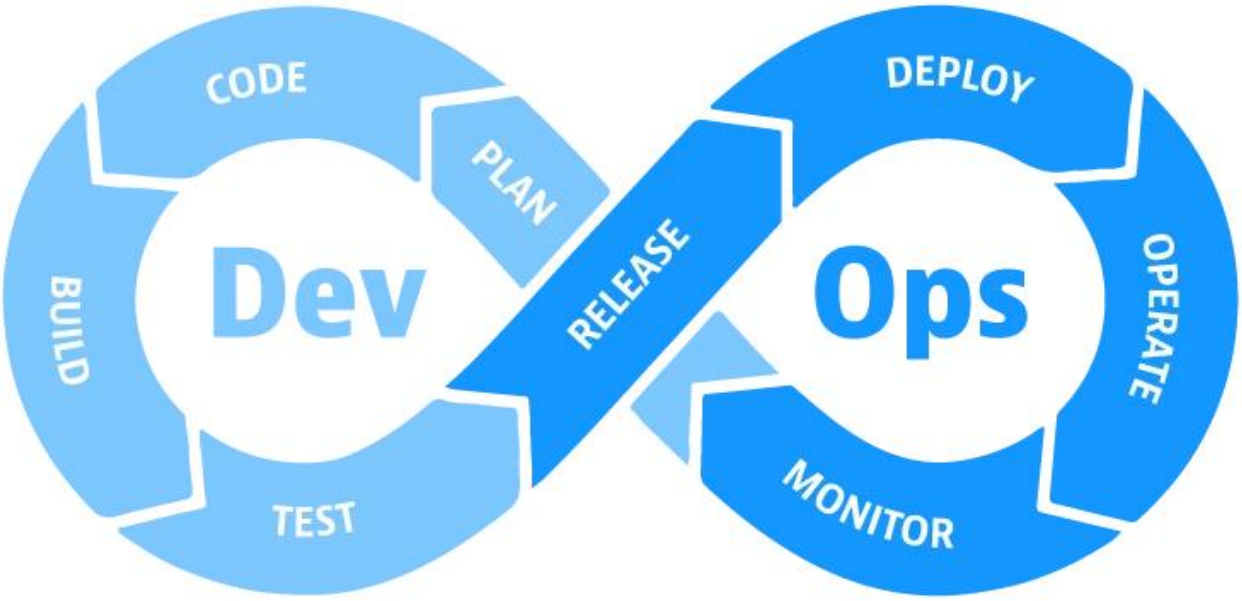
- Se um passo causa dor, execute-o com mais frequência e o quanto antes: Esse princípio incentiva a antecipação de problemas e a resolução deles o mais cedo possível. Se um determinado passo do processo de entrega de software é complicado ou propenso a erros, é recomendado realizá-lo com mais frequência e de forma contínua. Um exemplo disso é a prática de Integração Contínua, em que o código é integrado regularmente para evitar problemas de integração no futuro.

- Concluído significa pronto para entrega: Esse princípio visa eliminar ambiguidades e garantir que um trabalho esteja verdadeiramente concluído antes de ser considerado pronto para entrar em produção. Isso envolve não apenas a implementação do código, mas também a realização de testes adequados, documentação completa e integração com outros sistemas, se necessário. A conclusão deve ter uma semântica clara, indicando que o trabalho está 100% pronto para ser entregue.

- **Todos são responsáveis pela entrega do software:** Esse último princípio destaca a importância da colaboração e da quebra de silos entre as equipes de desenvolvimento e operações. Todos os membros envolvidos no ciclo de vida do software são responsáveis pela entrega bem-sucedida do produto. Isso promove uma cultura de colaboração, comunicação eficiente e compartilhamento de responsabilidades.

Ciclo de Vida

O ciclo de vida do DevOps é composto por **oito fases** que abrangem desde a concepção até o monitoramento contínuo. Vamos explorar cada uma dessas fases de forma mais didática:



1. **Planejamento (Plan):** Esta é a fase inicial do processo, na qual são definidos os objetivos, requisitos e cronograma do projeto. Os desenvolvedores e os operadores de infraestrutura trabalham juntos para garantir que os objetivos sejam alcançáveis e que a infraestrutura esteja preparada para suportar a nova versão do software.

2. **Codificação (Code):** Nesta fase, os desenvolvedores escrevem o código-fonte do novo software. O código deve ser escrito de acordo com as melhores práticas de desenvolvimento, de forma a ser seguro, eficiente e fácil de manter.

3. **Construção (Build):** Nesta fase, o código-fonte é compilado e empacotado em um formato executável. O processo de construção deve ser automatizado, de forma a garantir que o software seja construído sempre da mesma maneira.

4. **Teste (Test):** Nesta fase, o software é testado para garantir que atende aos requisitos e que não possui erros. Os testes devem ser realizados de forma automatizada, de forma a reduzir o tempo e o esforço necessários para testar o software.

5. **Lançamento (Release):** Nesta fase, o software é disponibilizado aos usuários. O lançamento deve ser feito de forma controlada, de forma a minimizar o impacto em caso de problemas.

processo de implantação deve ser automatizado, de forma a garantir que o software seja implantado com segurança e eficiência.

7. Operação (Operate): Nesta fase, o software é operado e mantido em produção. Os operadores de infraestrutura são responsáveis por garantir que o software esteja funcionando corretamente e que seja protegido de ataques.

8. Monitoramento (Monitor): Esta é a fase em que o software é monitorado para garantir que esteja funcionando corretamente e que esteja protegido de ataques. O monitoramento pode ser realizado por meio de ferramentas de monitoramento, que coletam dados sobre o desempenho do software.

Além dessas fases, o ciclo de vida do DevOps enfatiza o feedback contínuo, que ocorre em todas as etapas do processo. Ele envolve a coleta de feedback dos usuários, das equipes de operações e dos stakeholders para impulsionar melhorias contínuas no software e no processo de desenvolvimento.

Deployment Contínuo

Com a Integração Contínua (CI), novo código é frequentemente integrado no ramo principal do projeto, mesmo que não esteja pronto para entrar em produção. Isso permite que outros desenvolvedores tenham conhecimento da existência desse código e evitem conflitos futuros de integração. Por exemplo, é possível integrar uma versão preliminar de uma tela com uma interface ainda em desenvolvimento ou uma versão de uma função com problemas de desempenho.

No entanto, existe um passo adicional na automação proposta pelo DevOps, chamado de Deployment Contínuo (Continuous Deployment ou CD). A diferença entre CI e CD é simples, mas impactante: com o CD, cada novo commit no ramo principal é rapidamente colocado em produção, em questão de horas, por exemplo. O fluxo de trabalho com o CD é o seguinte:

O desenvolvedor desenvolve e testa o código em sua máquina local. Ele realiza um commit e o servidor de CI executa novamente o build e os testes de unidade. Algumas vezes por dia, o servidor de CI realiza testes mais abrangentes nos novos commits que ainda não foram colocados em produção. Isso inclui testes de integração, testes de interface e testes de desempenho. Se todos os testes forem aprovados, os commits são imediatamente colocados em produção, e os usuários já podem interagir com a nova versão do código.
Algumas vantagens do Deployment Contínuo são:

- Redução do tempo de entrega de novas funcionalidades. Em vez de esperar até que todas as funcionalidades estejam prontas para lançar uma nova versão, com o CD, as funcionalidades são liberadas assim que ficam prontas. Isso diminui o intervalo entre as releases, resultando em mais releases com um menor número de funcionalidades em cada uma delas.

Transformação das implantações em eventos não significativos. Não há mais um dia específico ou prazo final para lançar uma nova versão. Os prazos são uma fonte de estresse para os desenvolvedores e operadores de sistemas. Com o CD, as funcionalidades são implantadas assim que estiverem prontas, eliminando a pressão de prazos.

Com o CD, os desenvolvedores recebem feedback rapidamente, já que suas funcionalidades são colocadas em produção logo após serem concluídas. Isso evita que eles trabalhem por meses sem receber feedback sobre o sucesso ou falha de suas tarefas.

O favorecimento da experimentação e do desenvolvimento orientado por dados e feedback dos usuários é uma abordagem que visa aprimorar a criação e aprimoramento de produtos e serviços com base em informações concretas e observações reais. Com o CD, novas funcionalidades são rapidamente lançadas em produção, permitindo que os desenvolvedores recebam feedback dos usuários. Com base nesse feedback, é possível fazer ajustes na implementação ou até mesmo cancelar uma funcionalidade, se necessário.

No mundo real, várias empresas que desenvolvem sistemas web adotam o CD. Por exemplo, no Facebook, cada desenvolvedor faz em média 3,5 atualizações de software por semana, com uma média de 92 linhas de código adicionadas ou modificadas em cada atualização. Esses números demonstram que, para que o CD funcione bem, as atualizações de código devem ser pequenas. Para isso, os desenvolvedores precisam ter a habilidade de dividir qualquer tarefa de programação, mesmo que complexa, em partes menores que possam ser implementadas, testadas, integradas e entregues rapidamente. Essa abordagem permite que o CD funcione de maneira eficaz.

Entrega Contínua

A entrega contínua no DevOps é uma abordagem que visa tornar o processo de entrega de software mais **eficiente, confiável e rápida**. Ela envolve a **automação de várias etapas** do ciclo de vida do desenvolvimento de software, garantindo que as alterações feitas no código sejam compiladas, testadas e implantadas de forma automatizada.

Primeiramente, a entrega contínua inclui a **compilação automatizada do código-fonte**. Isso significa que, assim que um desenvolvedor conclui uma alteração no código, o sistema automaticamente compila o código para criar uma versão executável do software.

Em seguida, a entrega contínua envolve a execução de testes automatizados. Diferentes tipos de testes, como testes unitários, testes de integração e testes de aceitação, podem ser realizados de forma automatizada para verificar se o software funciona corretamente e atende aos requisitos estabelecidos.

Além disso, a entrega contínua inclui a criação de artefatos de implantação. Esses artefatos podem ser pacotes de software, imagens de contêiner ou qualquer outro formato necessário para implantar o software em diferentes ambientes.

Ademais, a entrega contínua abrange a implantação automatizada do software em ambientes de **teste, homologação e produção**. Com base nas configurações e regras definidas, o sistema pode implantar o software de forma automatizada em diferentes servidores ou infraestruturas, garantindo que a versão mais recente do software esteja disponível para uso.

A entrega contínua é suportada por pipelines de entrega, que são fluxos de trabalho automatizados que orquestram todas essas etapas. Esses pipelines são configurados para serem acionados automaticamente quando uma alteração de código é detectada e executam todas as etapas necessárias até a implantação do software.

Em alguns tipos de sistemas, como IDEs, navegadores web, aplicativos móveis e sistemas embutidos em hardware, o Deployment Contínuo (CD) não é adequado. Isso ocorre porque nesses casos os usuários não desejam ser constantemente interrompidos com notificações de novas versões. Imagine se toda vez que você abrisse o navegador no seu computador, fosse notificado sobre uma nova versão disponível, ou se seu celular o informasse diariamente sobre uma atualização do aplicativo de rede social que você usa. Além disso, alguns desses sistemas requerem um processo de instalação que não é transparente para os usuários, ao contrário das atualizações automáticas em um sistema web.

No entanto, para lidar com essas situações, pode-se adotar uma abordagem mais suave chamada de **Entrega Contínua (Continuous Delivery).** Nesse caso, cada commit feito pelos desenvolvedores

é considerado pronto para ser lançado em produção imediatamente. No entanto, a liberação efetiva para os usuários finais depende de uma autorização manual por parte de uma autoridade externa, como um gerente de projetos ou de releases. Essa decisão pode ser influenciada por fatores como estratégia da empresa ou demandas do mercado.

Na Entrega Contínua, cada alteração feita pelos desenvolvedores é considerada pronta para ser lançada imediatamente. Eles trabalham com essa mentalidade, mas a liberação para os usuários finais não acontece automaticamente. É necessária a aprovação de alguém que está no comando, como um gerente de projetos ou de lançamentos. Essa pessoa decide quando as alterações serão disponibilizadas aos usuários. Essa decisão pode ser influenciada por estratégias da empresa ou pelo mercado em que o sistema está inserido.

Para entender melhor, vamos diferenciar os conceitos: o Deployment é o processo de lançar uma nova versão do sistema para os usuários, tornando-a disponível para uso. Já o Delivery é o processo de disponibilizar uma nova versão do sistema para que seja liberada no deployment.

Dessa forma, a Entrega Contínua permite que as alterações sejam preparadas e prontas para serem lançadas a qualquer momento, mas a liberação efetiva para os usuários depende de uma aprovação manual. Isso permite um controle mais cuidadoso sobre as atualizações e considerações estratégicas antes de disponibilizá-las para os usuários finais.

No mundo real, podemos observar diferentes frequências de lançamentos de novas versões em sistemas que não são baseados na web. Isso nos dá exemplos concretos de como as empresas lidam com as atualizações de seus produtos. Vamos ver alguns exemplos:

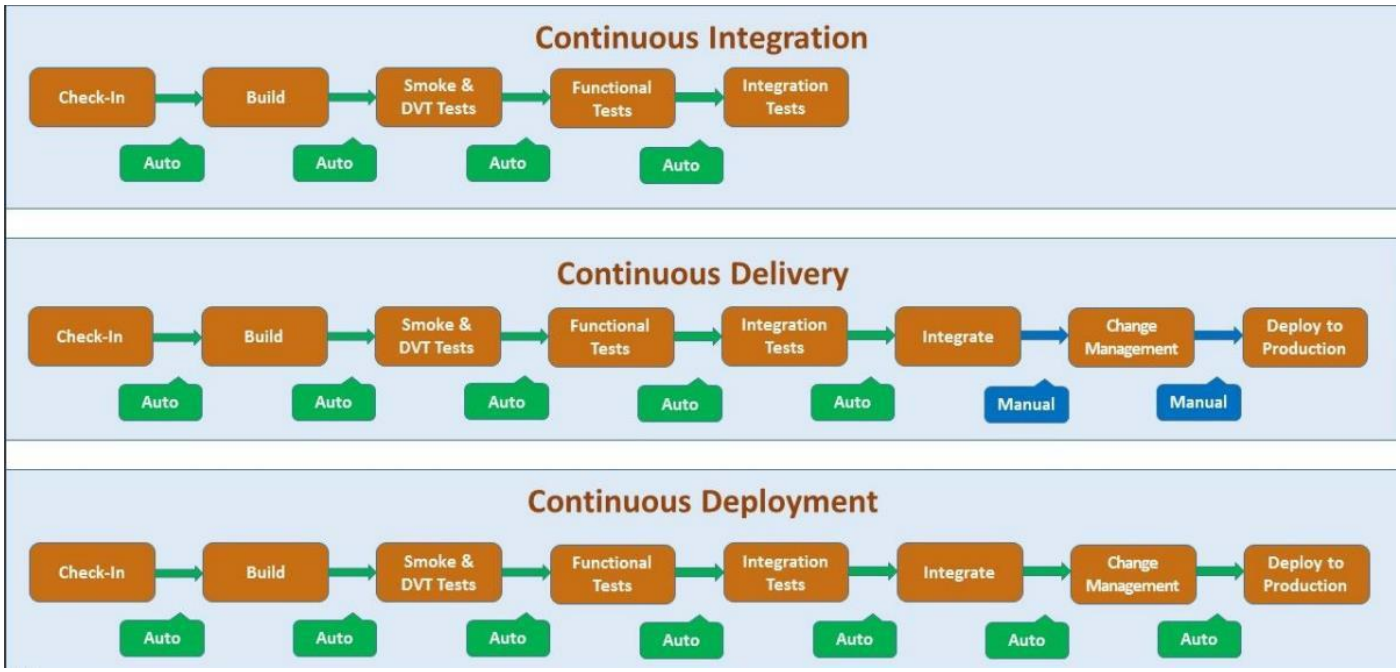
O Google, por exemplo, lança novas versões do navegador Chrome a cada seis semanas. Isso significa que a cada seis semanas os usuários podem esperar por atualizações e melhorias no navegador.

Anteriormente, a IDE Eclipse, que é uma ferramenta de desenvolvimento, costumava ter apenas uma nova versão por ano. No entanto, a partir de 2019, eles passaram a adotar um cronograma de lançamento mais frequente, com uma nova versão a cada 13 semanas. Isso permitiu que os desenvolvedores recebessem novas funcionalidades e melhorias com mais rapidez.

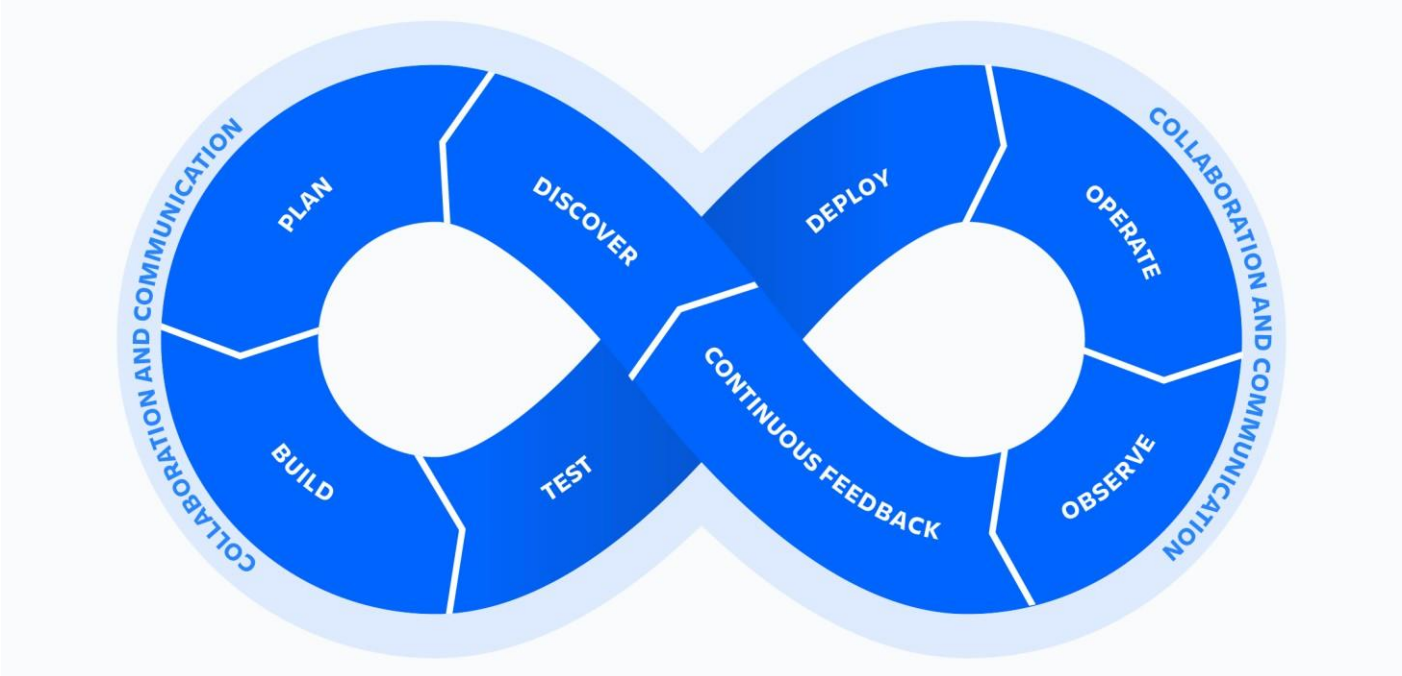
O Facebook, por sua vez, costumava atualizar sua versão para dispositivos Android a cada oito semanas. No entanto, recentemente eles reduziram esse tempo para apenas uma semana. Essa mudança tem como objetivo agradar os usuários, receber feedback mais rapidamente, manter os desenvolvedores motivados e se manter competitivo no mercado cada vez mais acelerado.

Esses exemplos mostram como as empresas estão se esforçando para lançar atualizações mais frequentes, a fim de melhorar seus produtos, atender às necessidades dos usuários e se manterem competitivas. Ao encurtar o tempo entre os lançamentos, eles podem garantir que novas

funcionalidades sejam entregues mais rapidamente, permitindo a obtenção de feedback valioso e aprimorando constantemente suas soluções.



O DevOps é mais do que apenas equipes de desenvolvimento e operações trabalhando juntas. É mais do que ferramentas e práticas. O DevOps é uma forma de pensar, uma mudança cultural, em que as equipes adotam novas formas de trabalhar.



Na cultura de DevOps, os desenvolvedores se aproximam dos usuários, obtendo uma compreensão melhor dos requisitos e necessidades deles. As equipes de operações se envolvem

no processo de desenvolvimento e adicionam requisitos de manutenção e necessidades do cliente. Ela também envolve aderir aos princípios essenciais a seguir, que ajudam as equipes de DevOps a oferecer aplicativos e serviços em um ritmo mais rápido e com maior qualidade do que as organizações que usam o modelo de desenvolvimento de software tradicional.

1. Automação: A automação é o uso de ferramentas e tecnologias para automatizar tarefas repetitivas e manuais ao longo do ciclo de vida do software. Isso inclui processos como compilação, teste, implantação e monitoramento. A automação é essencial para aumentar a eficiência, reduzir erros e acelerar o desenvolvimento e a entrega de software.

2. Colaboração ágil: A colaboração ágil é um princípio chave do DevOps que enfatiza a importância de equipes multidisciplinares trabalharem juntas de forma colaborativa e integrada. Isso envolve uma comunicação efetiva, compartilhamento de responsabilidades e conhecimento, e colaboração contínua ao longo do ciclo de vida do software. A colaboração ágil elimina silos organizacionais e promove a sinergia entre as equipes, resultando em uma entrega de software mais ágil e eficiente.

3. Produção contínua: A produção contínua, também conhecida como entrega contínua, é uma abordagem que busca entregar o software de forma consistente, confiável e repetível, pronta para ser colocada em produção. Isso envolve a automação do processo de implantação em ambientes de produção, permitindo implantações frequentes e seguras. A produção contínua reduz o tempo de espera para os usuários e permite um feedback mais rápido sobre o software.

4. Integração contínua: A integração contínua é uma prática em que as alterações de código são integradas de forma frequente e automatizada em um repositório compartilhado. O objetivo é identificar problemas de integração, conflitos e erros o mais cedo possível, garantindo que o software esteja sempre em um estado funcional. A integração contínua permite que as equipes trabalhem de forma colaborativa, integrando suas alterações de forma contínua e automatizada, o que reduz o risco de problemas decorrentes da integração tardia e melhora a eficiência do processo de desenvolvimento.

5. Ação centrada no cliente: A ação centrada no cliente é um princípio que destaca a importância de colocar o cliente no centro das atividades de desenvolvimento e operação de software. Isso envolve entender as necessidades e expectativas do cliente, coletar feedback regularmente e ajustar as prioridades e estratégias de desenvolvimento de acordo. Uma abordagem centrada no cliente garante que as equipes estejam desenvolvendo software que atenda às necessidades dos usuários finais e esteja alinhado com os objetivos do negócio.

Vamos analisar uma questão da FGV que aborda os princípios do DevOps. Nessa questão, é apresentado um cenário em que um Time de Desenvolvimento de Software (TDS) segue um

protocolo automatizado para gerar, testar e combinar pacotes de software gerados separadamente. Além disso, é mencionado que todo o software combinado precisa passar por um processo que inclui uma requisição formal ao Time de Operações (TO) de um Centro de Dados para executar um conjunto de testes, com o objetivo de verificar vulnerabilidades antes de entrarem produção. Vamos analisar.

(FGV – PGM - Niterói – 2023). Um Time de Desenvolvimento de Software (TDS) segue um protocolo automatizado para gerar, testar e combinar pacotes de software gerados separadamente. Todo software combinado precisa passar por um processo que inclui uma requisição formal ao Time de Operações (TO) de um Centro de Dados para executar um conjunto de testes, com o intuito de verificar vulnerabilidades no software antes de entrar em produção. Considerando os conceitos de DevOps e DevSecOps, o TDS e o TO estão falhando no princípio:

a) automação;
b) colaboração ágil;
c) produção contínua;
d) integração contínua;
e) ação centrada no cliente.

Comentários:

O gabarito é a letra A - automação. A falha no princípio da automação ocorre porque o processo descrito não é totalmente automatizado. Embora o Time de Desenvolvimento de Software (TDS) siga um protocolo automatizado para gerar, testar e combinar pacotes de software, há uma etapa em que é necessário fazer uma requisição formal ao Time de Operações (TO) para executar um conjunto de testes. Essa requisição formal indica que há uma intervenção manual no processo, o que vai contra o princípio da automação do DevOps e do DevSecOps.

As demais opções estão incorretas porque não estão diretamente relacionadas à falha mencionada. Colaboração ágil (opção B): Embora a colaboração entre o Time de Desenvolvimento de Software (TDS) e o Time de Operações (TO) seja mencionada, não há indicação de uma falha específica nessa colaboração.
Produção contínua (opção C): O texto menciona que os pacotes de software são combinados separadamente e passam por testes antes de entrar em produção. Portanto, não há uma falha específica no princípio da produção contínua.
Integração contínua (opção D): Embora o texto mencione que os pacotes de software são combinados, não há informações suficientes para determinar se há uma falha no princípio da integração contínua.
Ação centrada no cliente (opção E): O texto não menciona diretamente a relação com o cliente, portanto não é possível afirmar que há uma falha no princípio da ação centrada no cliente. (Gabarito: Letra A)

Pessoal, vimos que a situação descrita se refere ao princípio de "Automação" no contexto de DevOps. No caso apresentado, o Time de Desenvolvimento de Software (TDS) segue um protocolo automatizado para gerar, testar e combinar pacotes de software gerados separadamente. A automação é o uso de ferramentas e tecnologias para automatizar tarefas repetitivas e manuais ao longo do ciclo de vida do software. Nesse caso, o processo de geração, teste e combinação de pacotes de software é realizado de forma automatizada, o que contribui para aumentar a eficiência, reduzir erros e acelerar o desenvolvimento e a entrega de software.

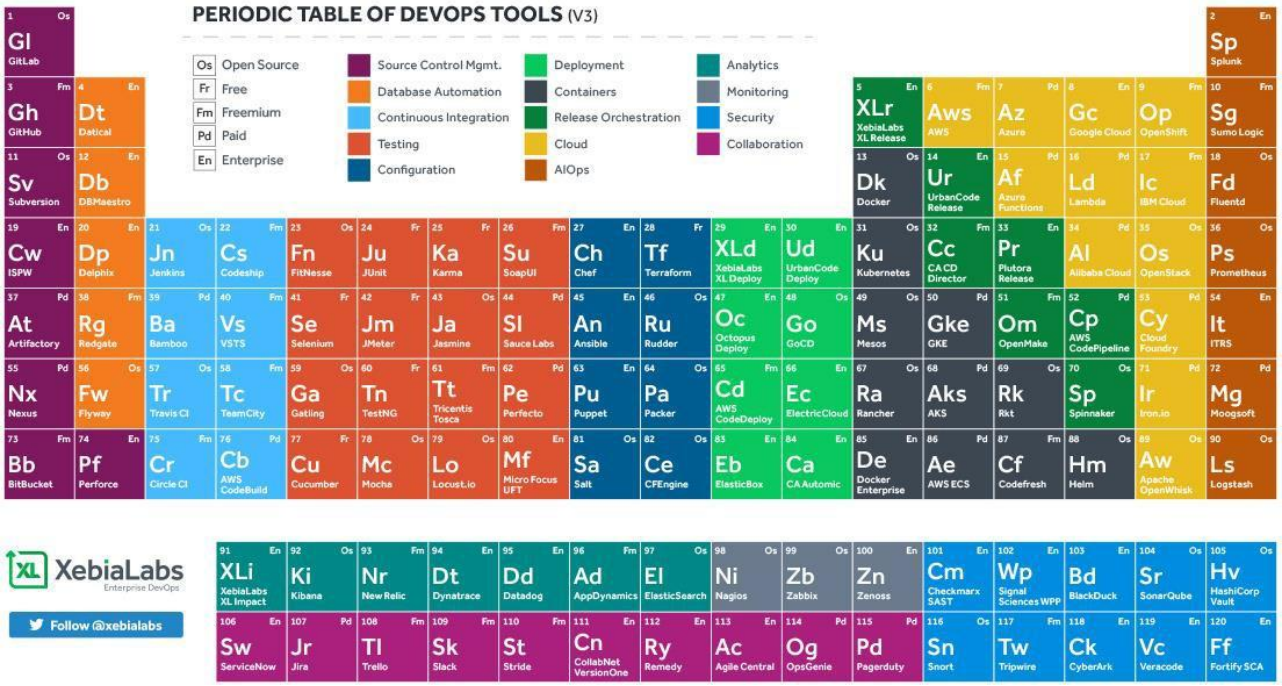
Além disso, é mencionado que o software combinado precisa passar por um processo de teste, no qual uma requisição formal é enviada ao Time de Operações (TO) de um Centro de Dados para executar um conjunto de testes. Essa abordagem de automatizar o processo de teste também está alinhada com o princípio de automação, pois ajuda a garantir a qualidade do software de forma consistente e confiável.

Ferramentas de DevOps

A Tabela Periódica de Ferramentas DevOps[1] é um recurso abrangente que mostra várias ferramentas usadas no cenário DevOps, ela é SENSACIONALMENTE DIDÁTICA. Ela fornece uma visão categorizada dessas ferramentas. A Tabela Periódica de Ferramentas DevOps organiza AS ferramentas em diferentes categorias, facilitando a compreensão e escolha das corretas para tarefas específicas.

A tabela inclui 14 categorias, cada uma representando um aspecto diferente do DevOps. Essas categorias incluem Integração Contínua, Entrega Contínua, Testes, Gerenciamento de Configuração, Orquestração de Infraestrutura, Containerização, Monitoramento, Gerenciamento de Logs, Colaboração, Segurança, Gerenciamento de Banco de Dados, Gerenciamento de Versões e Automação de Builds.

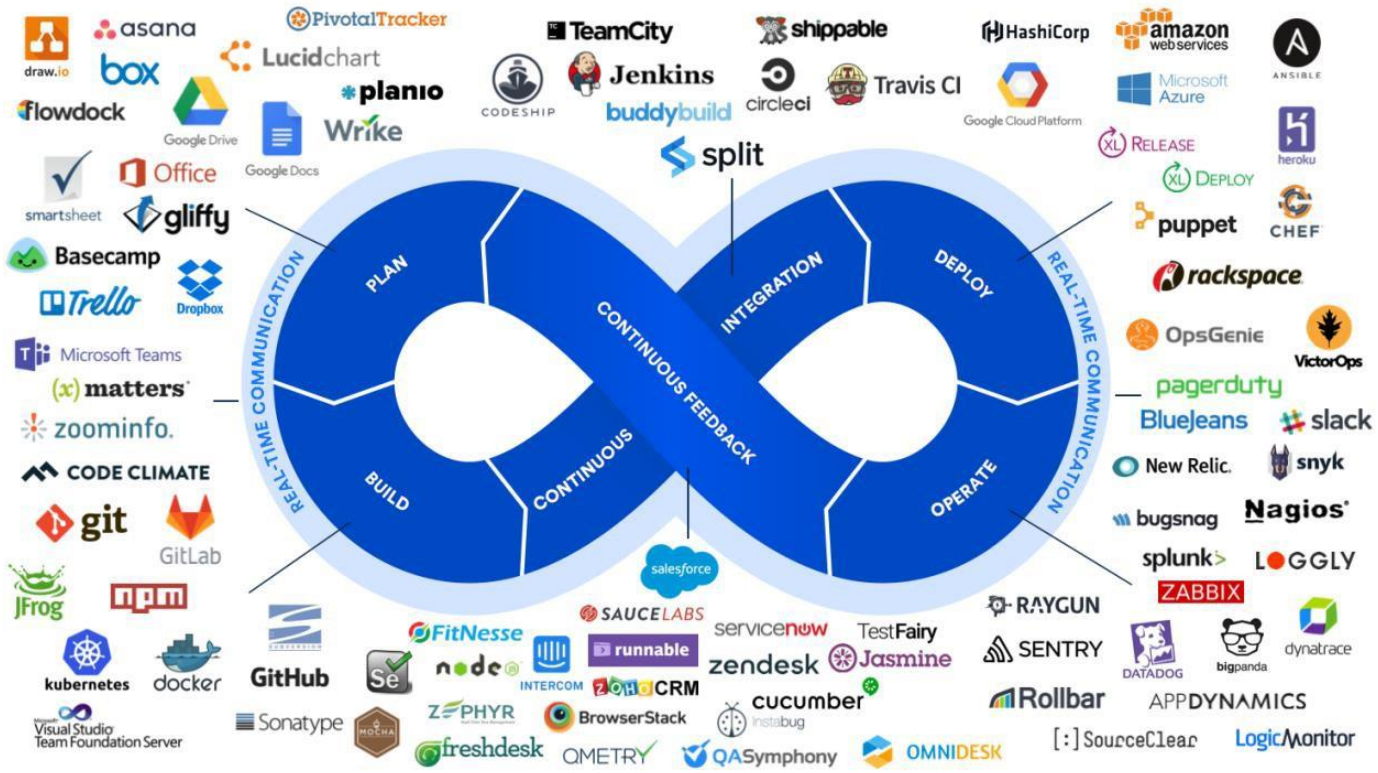
Dentro de cada categoria, você encontrará ferramentas populares amplamente usadas na indústria. A tabela fornece uma representação visual dessas ferramentas, ajudando os profissionais de TI a se manterem atualizados com as opções mais recentes disponíveis.



[1] https://xebialabs.com/wp-content/uploads/files/infographics/periodic-table-of-devops-tools-v3-1.pdf
https://xebialabs.com/periodic-table-of-devops-tools/

As ferramentas são fundamentais para automatizar o DevOps. Entregas de qualidade com ciclos curtos de tempo exigem um alto grau de automação. Portanto, escolher as ferramentas certas para o seu ambiente ou projeto é importante quando você adota o DevOps.

Pessoal, abaixo temos a lista mais completa que encontrei com as ferramentas possíveis de utilização no ciclo DevOps.



Colaboração

A etapa de colaboração no DevOps é uma das fases essenciais desse processo, que envolve a interação e cooperação entre as equipes de desenvolvimento e operações. Nessa etapa, as equipes trabalham em conjunto para compartilhar informações, alinhar objetivos, tomar decisões conjuntas e garantir uma comunicação eficiente ao longo de todo o ciclo de vida do software.

A colaboração no DevOps busca superar as barreiras tradicionais entre as equipes de desenvolvimento e operações, promovendo uma cultura de colaboração, transparência e responsabilidade compartilhada. Através dessa colaboração, as equipes podem identificar e solucionar problemas de forma mais rápida, garantir a entrega contínua de software de qualidade e responder de maneira ágil às necessidades dos usuários.

Nessa etapa, são utilizadas diversas ferramentas de colaboração, como JIRA, Trello, Basecamp, Asana, Microsoft Teams, Slack e muitas outras, que permitem a comunicação eficiente, o compartilhamento de informações, o gerenciamento de tarefas e a coordenação das atividades

entre as equipes. A colaboração no DevOps é um princípio fundamental para promover uma cultura de trabalho colaborativa, em que os membros das equipes se apoiam mutuamente, compartilham conhecimentos, experiências e responsabilidades, visando alcançar os objetivos comuns de entrega contínua, qualidade e eficiência na produção de software. A colaboração é algo significativo para todas as aplicações. Uma aplicação ou software não é muito útil se for utilizado apenas para um único propósito. Ao invés disso, se o seu software colaborar com outros softwares presentes no mercado, então ele se torna útil para ambos. Portanto, as principais ferramentas através das quais você pode colaborar com o seu software são as seguintes:

- JIRA: O JIRA é uma ferramenta de gerenciamento de projetos que permite o acompanhamento de tarefas, atribuição de responsabilidades, definição de prazos e monitoramento do progresso do projeto. Ele auxilia na organização das atividades e facilita a comunicação entre os membros da equipe.

- Trello: O Trello é uma ferramenta de gerenciamento de projetos baseada em quadros virtuais. Ele permite a criação de listas e cartões para representar tarefas e atividades. Os membros da equipe podem mover os cartões entre as listas para indicar o status atual de cada tarefa, facilitando o acompanhamento do progresso.

- Slack: O Slack é uma ferramenta de comunicação em equipe que oferece recursos avançados de mensagens instantâneas, chamadas de áudio e vídeo, compartilhamento de arquivos e integração com outras ferramentas. Ele permite que os membros da equipe se comuniquem de forma eficiente e colaborativa, facilitando a troca de informações e o trabalho em conjunto.

Gerenciamento de Código Fonte

Ao começarmos a desenvolver um aplicativo usando a abordagem DevOps, um dos primeiros passos é construir o código. Como cada aplicativo possui um código em execução que precisa ser atualizado conforme necessário, é importante gerenciar o código-fonte. As ferramentas de gerenciamento de código-fonte fornecem recursos para mostrar qual usuário fez alterações em determinado momento. As ferramentas mais populares nesse segmento são as seguintes:

- GitHub: O GitHub é uma plataforma de hospedagem de repositórios de controle de versão. Ele permite que os desenvolvedores colaborem em projetos, realizem o controle de versão do código-fonte e gerenciem as alterações feitas por diferentes membros da equipe. O GitHub oferece recursos de rastreamento de problemas, revisão de código e integração com outras ferramentas de desenvolvimento.

- Subversion: O Subversion (também conhecido como SVN) é um sistema de controle de versão centralizado amplamente utilizado. Ele permite que várias pessoas trabalhem em um projeto simultaneamente, rastreiem as alterações feitas no código e revertam para versões

anteriores, se necessário. O Subversion oferece recursos de ramificação e mesclagem, facilitando o gerenciamento de diferentes fluxos de trabalho de desenvolvimento.

- Artifactory: O Artifactory é um repositório de artefatos que gerencia e armazena os componentes de software criados durante o desenvolvimento. Ele permite o gerenciamento centralizado de pacotes, bibliotecas, dependências e outros artefatos necessários para construir e implantar um aplicativo. O Artifactory oferece recursos de gerenciamento de versões, controle de acesso e integração com ferramentas de automação de build e implantação.

- Nexus: O Nexus é outro exemplo de repositório de artefatos usado para armazenar e distribuir componentes de software. Ele oferece recursos de gerenciamento de dependências, controle de acesso, geração de relatórios e proxy para repositórios remotos. O Nexus é amplamente utilizado na integração contínua e entrega contínua para garantir que as dependências do projeto estejam disponíveis e atualizadas.

- Bitbucket: O Bitbucket é uma plataforma de hospedagem de repositórios de controle de versão, semelhante ao GitHub. Ele permite que os desenvolvedores armazenem, colaborem e controlem as versões de seu código-fonte. O Bitbucket suporta tanto repositórios Git quanto Mercurial e oferece recursos como revisão de código, integração com outras ferramentas de desenvolvimento e recursos avançados de gerenciamento de equipe.

Automação de Banco de Dados

Bancos de dados desempenham um papel fundamental em qualquer tipo de aplicação. No entanto, é praticamente impossível para os desenvolvedores realizar tarefas administrativas nos bancos de dados com frequência. Portanto, a automação de banco de dados consiste na aplicação de processos de atualização e administração não supervisionados em bancos de dados. Com esse tipo de automação, é possível reduzir erros nas operações, melhorar a velocidade e aumentar a confiabilidade. Algumas das ferramentas populares utilizadas para esse propósito são as seguintes:

- Datical: O Datical é uma ferramenta de automação de banco de dados que auxilia no gerenciamento e na implementação de alterações no banco de dados de forma segura e eficiente. Ele permite que as equipes de desenvolvimento e operações colaborem no controle de versões do banco de dados, evitando erros e facilitando a entrega contínua.

- DBMaestro: O DBMaestro é uma ferramenta de gerenciamento de alterações de banco de dados que ajuda as equipes a controlar e gerenciar as mudanças feitas nos bancos de dados. Ele oferece recursos de controle de versão, rastreamento de alterações, implantação automatizada e conformidade com regulamentos de segurança.

- Delphix: O Delphix é uma plataforma de automação de dados que permite a criação de cópias virtuais de bancos de dados para fins de desenvolvimento, teste e análise. Ele ajuda as equipes a economizar tempo e recursos, fornecendo dados virtualizados sob demanda, sem a necessidade de cópias físicas dos bancos de dados.

- Redgate: A Redgate fornece uma variedade de ferramentas para o gerenciamento e a automação de bancos de dados, incluindo ferramentas para controle de versão, comparação e sincronização de esquemas, monitoramento de desempenho e segurança. Essas ferramentas ajudam as equipes a garantir a integridade e a eficiência dos bancos de dados.

- Flyway: O Flyway é uma ferramenta de migração de banco de dados que permite que as equipes gerenciem e versionem as alterações feitas nos esquemas dos bancos de dados. Ele fornece um sistema simples e poderoso para rastrear, aplicar e reverter as migrações do banco de dados, garantindo a consistência e a confiabilidade dos dados.

Integração Contínua

Imagine que você está trabalhando em um projeto de desenvolvimento de software juntamente com outros colegas de equipe. Cada um de vocês está trabalhando em diferentes partes do código, adicionando novas funcionalidades ou corrigindo bugs.

Agora, suponha que todos vocês trabalhem de forma isolada durante dias ou semanas sem compartilhar o código entre si. Quando chegar a hora de integrar todas as partes do código, vocês podem enfrentar grandes problemas. Pode haver conflitos entre as alterações realizadas por diferentes pessoas, o que torna difícil e demorado resolver esses conflitos manualmente.

É aí que entra a Integração Contínua. Essa prática sugere que, em vez de esperar até o final do projeto para integrar todo o código, vocês devem realizar integrações frequentes ao longo do processo de desenvolvimento. Isso significa que cada vez que um desenvolvedor terminar uma pequena tarefa ou implementar uma funcionalidade, ele deve integrar seu código ao repositório compartilhado o mais rápido possível. Dessa forma, vocês evitam acumular grandes quantidades de código não integrado. Quando o código é integrado com frequência, é mais fácil identificar e resolver problemas de integração. Além disso, as alterações feitas por diferentes desenvolvedores são combinadas regularmente, o que ajuda a detectar conflitos e erros mais cedo no processo, facilitando a sua correção.

Paolla, mas quão rápido é "integrar seu código ao repositório compartilhado o mais rápido possível"? A velocidade de integração do código ao repositório compartilhado na prática de Integração Contínua pode variar, mas o objetivo é realizar integrações frequentes e regulares paraobter os benefícios dessa abordagem.

Kent Beck, um dos defensores do Extreme Programming (XP) e da Integração Contínua, sugere que os **desenvolvedores devem integrar e testar seu código em intervalos menores do que algumas horas**. Ele argumenta que a **duração da tarefa de integração é imprevisível** e pode levar mais tempo do que a tarefa original de codificação. Portanto, quanto mais tempo for necessário para integrar o código, maiores e mais imprevisíveis serão os custos associados. Outros especialistas, como Martin Fowler, mencionam que pelo menos uma integração por dia por desenvolvedor é um limite mínimo para considerar que uma equipe está aplicando a Integração Contínua.

Essas referências sugerem que a integração contínua deve acontecer com uma frequência significativa, mas não necessariamente a cada poucos minutos ou segundos. O principal objetivo é evitar longos períodos de trabalho isolado e garantir que as alterações de código sejam integradas e testadas em intervalos regulares, minimizando a chance de conflitos e problemas de integração. O tempo exato entre as integrações pode variar dependendo das necessidades e da complexidade do projeto, bem como das preferências da equipe. O importante é encontrar um ritmo que permita uma colaboração contínua, detecção precoce de problemas e resolução eficiente de conflitos. Vejamos essa citação de Kent Beck:

> Você deve integrar e testar o seu código em intervalos menores do que algumas horas. Programação em times não é um problema do tipo dividir-e-conquistar. Na verdade, é um problema que requer **dividir, conquistar e integrar**. A duração de uma tarefa de integração é imprevisível e pode facilmente levar mais tempo do que a tarefa original de codificação. Assim, quanto mais tempo você demorar para integrar, maiores e mais imprevisíveis serão os custos.

Você deve integrar e testar o seu código em intervalos menores do que algumas horas. Programação em times não é um problema do tipo dividir-e-conquistar. Na verdade, é um problema que requer dividir, conquistar e integrar. A duração de uma tarefa de integração é imprevisível e pode facilmente levar mais tempo do que a tarefa original de codificação. Assim, quanto mais tempo você demorar para integrar, maiores e mais imprevisíveis serão os custos.

A Integração Contínua também envolve a execução automatizada de testes em cada integração. Assim, vocês garantem que o código integrado seja testado continuamente, identificando possíveis problemas o mais cedo possível. Ao adotar a Integração Contínua, vocês reduzem os riscos de erros e conflitos tardios, economizando tempo e esforço. Além disso, ela promove uma colaboração mais estreita entre os membros da equipe, evitando que o trabalho seja desenvolvido em silos isolados.

Boas Práticas para Uso de CI

Quando falamos de CI, estamos nos referindo a um processo de desenvolvimento de software em que o código é constantemente atualizado e integrado na ramificação principal, também

conhecida como "master". O objetivo é garantir que o código esteja sempre funcionando corretamente, sem erros ou bugs, mesmo com as atualizações mais recentes. Existem algumas práticas recomendadas para garantir o bom funcionamento da CI:

- Build Automatizado: O processo de build consiste em compilar todos os arquivos de um sistema, criando uma versão executável. Para a integração contínua, é fundamental que esse processo seja automatizado, ou seja, sem intervenção manual. Isso ajuda a garantir que o build seja rápido e eficiente, pois será executado repetidamente. Autores especializados sugerem que o tempo de execução de um build não ultrapasse 10 minutos.

- Testes Automatizados: Além de garantir que o código compile sem erros, é importante verificar se o sistema continua funcionando como esperado após cada atualização. Por isso, é recomendado ter uma ampla cobertura de testes automatizados, especialmente testes de unidade. Esses testes ajudam a identificar possíveis problemas e bugs nas funcionalidades do sistema, garantindo sua qualidade.

- Servidores de CI são ferramentas que auxiliam na implementação da integração contínua em um projeto de desenvolvimento de software. Eles desempenham um papel fundamental na automatização dos processos de build e testes.

Vamos entender passo a passo como os Servidores de CI funcionam:

1. Após um novo commit: Quando um desenvolvedor realiza um novo commit no sistema de controle de versões, como o Git, o servidor de CI é notificado.

2. Clone do repositório: O servidor de CI realiza uma cópia do repositório do projeto, isso é conhecido como "clonar o repositório". Ele faz isso para obter a versão atualizada do código em que o commit foi realizado.

3. Execução do build completo: Após clonar o repositório, o servidor de CI inicia o processo de build completo do sistema. Esse processo envolve a compilação e a construção de todos os arquivos do projeto, resultando em uma versão executável do software. O objetivo é garantir que o código seja compilado corretamente, sem erros.

4. Execução dos testes: Após o build completo, o servidor de CI executa todos os testes automatizados configurados para o projeto. Isso inclui testes de unidade, integração, aceitação e outros, dependendo das necessidades do projeto. O objetivo é verificar se o sistema continua com o comportamento esperado e se não houve introdução de erros.

5. Notificação do usuário: Após a execução do build e dos testes, o servidor de CI notifica o usuário sobre os resultados. Isso pode ser feito por meio de mensagens, e-mails, notificações no sistema, entre outros meios de comunicação. Essa comunicação serve para

informar ao desenvolvedor sobre o status do processo de CI, indicando se o build e os testes foram bem-sucedidos ou se ocorreram falhas.

(CESPE – Petrobras – 2022). Julgue o item subsecutivo, relativos a DevOps e notação BPMN.

No DevOps, a integração contínua possui como uma de suas atividades a realização de testes; a fim de se obter os benefícios esperados convém automatizar os testes para poder executá-los para cada alteração feita no repositório principal.

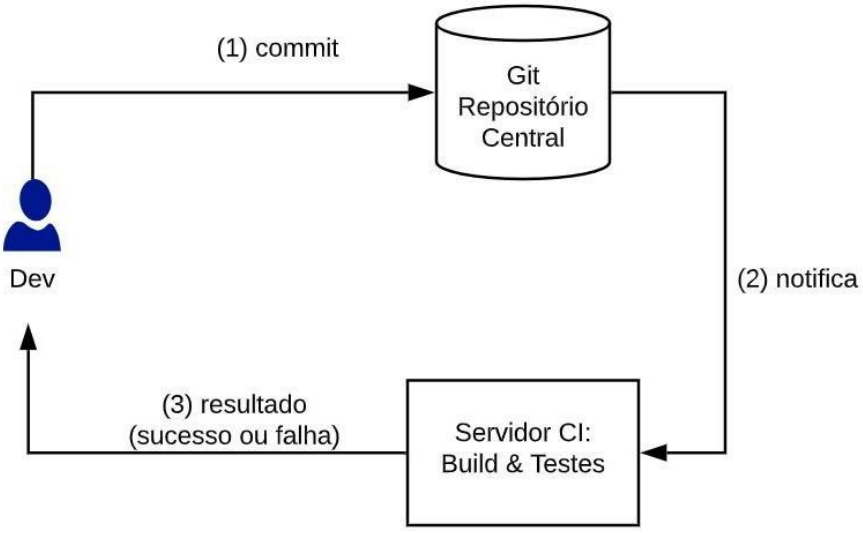
Comentários:

Na prática, muitas organizações utilizam sistemas de controle de versão, como o Git, para automatizar processos de DevOps. Esses sistemas permitem que equipes de desenvolvedores trabalhem em conjunto em um mesmo projeto, realizando alterações no código-fonte de forma simultânea. Com a automação, é possível evitar conflitos e preservar o histórico do código.

Um exemplo dessa automação ocorre na integração contínua. Sempre que são feitas alterações no repositório principal do código-fonte, os testes automatizados são executados. Isso garante que, a cada mudança realizada, os testes necessários sejam executados de forma rápida e eficiente.

Dessa forma, a automação dos testes permite que a equipe de desenvolvimento mantenha a qualidade do software, identificando erros e falhas de forma ágil. Além disso, ao automatizar os testes, é possível ter um feedback constante sobre o estado do código, garantindo que ele permaneça funcional e livre de problemas. Essa prática contribui para a eficiência do processo de desenvolvimento, uma vez que os testes são executados de maneira automática e contínua, sem a necessidade de intervenção manual a cada alteração no código-fonte. (Gabarito: Correto)

Os Servidores de CI são ferramentas poderosas que automatizam processos essenciais para garantir a qualidade do código e a estabilidade do sistema. Eles permitem a detecção precoce de problemas e a identificação de erros antes que sejam integrados à ramificação principal do projeto.



O principal objetivo de um servidor de integração contínua é evitar a integração de código problemático, seja em relação ao processo de compilação ou ao comportamento do sistema. Quando ocorre uma falha no build, é comum dizer que o build quebrou. Às vezes, o build pode ter sido bem-sucedido na máquina do desenvolvedor, mas falha ao ser executado no servidor de CI. Isso pode acontecer se o desenvolvedor esquecer de fazer o commit de algum arquivo ou se houver dependências incorretas. Por exemplo, o código pode ter sido compilado e testado na máquina do desenvolvedor usando a versão 2.0 de uma biblioteca, enquanto o servidor de CI utiliza a versão 1.0.

Quando o servidor de CI notifica o desenvolvedor de que seu código não passou nos testes ou quebrou o build, é importante que ele pare o que está fazendo e corrija o problema imediatamente. Isso é crucial porque um build quebrado afeta o trabalho dos outros desenvolvedores, que não conseguirão mais compilar ou executar o sistema. Diz-se que a correção de um build quebrado é a maior prioridade em uma empresa de software. No entanto, em alguns casos, a solução pode ser simplesmente reverter o código para a versão anterior ao commit problemático.

A fim de manter a coerência com os princípios da Integração Contínua, é recomendado que os desenvolvedores aguardem o resultado do servidor de CI antes de avançarem para a próxima tarefa de programação. Isso significa que eles não devem iniciar a escrita de novo código até que tenham certeza de que o commit mais recente tenha passado pelo serviço de CI. Além disso, é importante evitar iniciar outras atividades significativas, como reuniões, almoços ou ir para casa, antes de receber o resultado do servidor de CI.

Existem várias opções de servidores de integração contínua disponíveis no mercado. Alguns são oferecidos como serviços independentes, sendo gratuitos para repositórios de código aberto, mas requerendo pagamento para repositórios privados de empresas. Portanto, se você possui um repositório aberto no GitHub, há mais de uma opção gratuita para ativar um serviço de CI nele.

Uma dúvida comum é se a Integração Contínua é compatível com o uso de branches. De acordo com a definição da Integração Contínua, a resposta é sim, desde que os branches sejam integrados frequentemente ao ramo principal (geralmente o "master"), preferencialmente diariamente. Em outras palavras, a Integração Contínua não é incompatível com o uso de branches, mas apenas

com branches de longa duração. Por exemplo, Martin Fowler observa que o uso de branches específicos para funcionalidades pode ser adotado em conjunto com a Integração Contínua, desde que esses branches sejam integrados regularmente.

> Na maioria das vezes, branches de funcionalidades constituem uma abordagem incompatível com CI. Um dos princípios de CI é que todos devem enviar commits para a linha de desenvolvimento principal diariamente. Então, a não ser que os branches de funcionalidades durem menos do que um dia, eles são um animal diferente de CI. É comum ouvir desenvolvedores dizendo que eles estão usando CI porque eles rodam builds automáticos, talvez usando um servidor de CI, após cada commit. Isso pode ser chamado de building contínuo e pode ser uma coisa boa. Porém, como não há integração, não podemos chamar essa prática de CI.

Programação em Pares

O Pair Programming, ou Programação em Pares, pode ser considerado uma forma contínua de revisão de código. Nessa prática, um desenvolvedor trabalha em conjunto com outro, sentando ao lado dele, revisando e colaborando na escrita de cada trecho de código. Assim como a Integração Contínua (CI), é recomendado utilizar o Pair Programming em conjunto com a CI. No entanto, seu uso não é obrigatório.

No Pair Programming, o código é revisado imediatamente durante a sessão de programação, garantindo que haja um controle contínuo de qualidade. Dessa forma, erros podem ser identificados e corrigidos rapidamente, antes mesmo do commit no repositório principal. Essa abordagem ajuda a minimizar os custos associados à revisão de código após a integração, quando os problemas podem ser mais complexos de solucionar.

No entanto, é possível realizar a revisão de código após o commit ser realizado no repositório principal. Nesse caso, os custos e esforços necessários para a revisão podem ser maiores, uma vez que o código já foi integrado e pode afetar outros desenvolvedores. Portanto, embora seja possível revisar o código após a integração, recomenda-se a adoção do Pair Programming para garantir uma revisão contínua e colaborativa, reduzindo os riscos de erros e melhorando a qualidade do código desde o início.

A Integração Contínua é o núcleo do Ciclo de Vida do DevOps, pois todos os membros de uma equipe coordenam seu trabalho com frequência. Cada integração é verificada por um processo automatizado para identificar a integração o mais rápido possível. Aqui, você só precisa lembrar que é necessário escolher um método confiável de correspondência para garantir que erros sejam encontrados muito mais cedo no pipeline de CI/CD. Algumas das conhecidas ferramentas de integração contínua são as seguintes:

- Jenkins: O Jenkins é uma ferramenta de integração contínua que automatiza o processo de construção, testes e implantação de software. Ele permite que as equipes de desenvolvimento integrem as alterações de código em um repositório compartilhado, onde são automaticamente construídas, testadas e implantadas em um ambiente de desenvolvimento ou produção.

- Codeship: O Codeship é uma plataforma de integração e entrega contínua (CI/CD) baseada na nuvem. Ele permite que as equipes de desenvolvimento automatizem o processo de integração de código, construção, testes e implantação em diversos ambientes. Com o Codeship, é possível criar pipelines de entrega contínua para garantir a qualidade e eficiência do processo de desenvolvimento de software.

- AWS CodeBuild: O AWS CodeBuild é um serviço gerenciado da Amazon Web Services (AWS) que compila e testa o código-fonte de aplicativos de forma automatizada. Ele permite que as equipes de desenvolvimento construam, testem e implantem aplicativos de forma rápida e confiável na nuvem da AWS. O AWS CodeBuild é altamente escalável e integra-se perfeitamente com outras ferramentas e serviços da AWS.

Build (Construção)

Nessa fase do DevOps, as ferramentas devem suportar fluxos de trabalho rápidos. Aqui discutimos dois tipos de ferramentas. As ferramentas de construção ajudam a alcançar iterações rápidas, reduzindo tarefas manuais demoradas. As ferramentas de integração contínua (CI) mesclam o código de todos os desenvolvedores e verificam se há código com problemas, melhorando a qualidade do software.

Ferramentas de construção. A automação de construção é crucial para o ambiente de DevOps - tanto para projetos pequenos quanto grandes. Com o desenvolvimento ágil e fluido, as ferramentas de construção também se tornaram ferramentas para gerenciar o ciclo de vida do desenvolvimento de software e do serviço, o que envolve compilar código, gerenciar dependências, gerar documentação, executar testes ou implantar um aplicativo em diferentes ambientes.

O Apache Ant se tornou a ferramenta padrão para a construção de aplicativos, especialmente em projetos de código aberto quando é necessário compilar a partir das fontes. A sintaxe simples usada para definir as tarefas a serem executadas reflete a principal vantagem e desvantagem do Ant. É fácil de entender, mas é verboso até mesmo para tarefas simples, porque usa XML, que não é uma linguagem de programação procedural.

O Maven visa resolver alguns dos problemas do Ant. Ele também usa XML para definir o processo de construção. Nesse caso, as declarações definem a natureza do projeto em vez das tarefas que o constroem. O Maven usa o Project Object Model, que define uma maneira uniforme de construir

sistemas. Isso tem a vantagem de definir um padrão no início de um projeto, mas às vezes causa inflexibilidade quando operações personalizadas precisam ser definidas. O gerenciamento de dependências também ajuda a automatizar o processo de construção, mas às vezes cria problemas devido às dependências externas automáticas.

Rake e Gradle tentam resolver os problemas do Ant e do Maven, oferecendo uma ferramenta baseada em linguagem de programação para construir aplicativos. Com o Rake, as linhas de código são escritas em Ruby; com o Gradle, a configuração usa uma linguagem específica de domínio (DSL) baseada em Groovy.

Ferramentas de integração contínua: A integração contínua (CI) é um conceito ágil fundamental. As ferramentas de CI integram o trabalho dos desenvolvedores o mais cedo e o mais frequentemente possível. Dessa forma, o sistema é constantemente testado. Adotar a CI nem sempre é fácil e direto. Por exemplo, no desenvolvimento de sistemas embarcados, os testes são desafiadores porque nem sempre é possível construir e testar no hardware de destino e, portanto, deve ser simulado. Além disso, as limitações de hardware afetam a velocidade dos testes e, consequentemente, o tempo do ciclo.

O Jenkins é um sistema de código aberto baseado em Java e uma das ferramentas mais utilizadas, por isso possui muitas opções de plug-ins. Devido à sua popularidade, é fácil encontrar suporte em sua grande base de usuários. No entanto, sua interface de usuário está desatualizada e é menos atraente do que as interfaces das outras ferramentas.

O TeamCity, desenvolvido pela ITBrains, é baseado em Java e, portanto, tem bom suporte para Java, assim como o Jenkins. Ao contrário do Jenkins, ele também oferece bom suporte para .NET. Embora seja licenciado, possui uma edição gratuita para projetos pequenos.

O Bamboo é da Attlasian e se integra bem com outras ferramentas da Attlasian. Portanto, oferece vantagens se você já está usando ferramentas da Attlasian.

Implantação

Durante essa fase, a mudança mais importante é tratar a infraestrutura como código. Com essa abordagem, a infraestrutura pode ser compartilhada, testada e controlada por versões. Desenvolvimento e produção compartilham uma infraestrutura homogênea, reduzindo problemas e bugs devido a diferentes configurações de infraestrutura.

O DevOps promove a automação desde a aplicação até a infraestrutura. Em comparação com o provisionamento manual de infraestrutura, ferramentas de gerenciamento de configuração podem reduzir a complexidade do provisionamento e da manutenção de configurações de produção, permitindo a recriação do sistema de produção nas máquinas de desenvolvimento. Essas

ferramentas são um grande facilitador do DevOps, especialmente à medida que a arquitetura passa de um bloco monolítico de software para uma abordagem de microsserviços.

Para alcançar a entrega contínua para sistemas embarcados, os dispositivos devem estar conectados, portanto, a comunicação máquina a máquina e uma arquitetura de Internet das Coisas são necessárias. A entrega contínua melhorará o tempo de lançamento no mercado e permitirá práticas ágeis com feedback rápido dos consumidores. No entanto, para aplicativos críticos em que a falha não é uma opção, como aplicativos médicos ou aeroespaciais, uma abordagem mais conservadora é preferível. As seguintes ferramentas ajudarão no ciclo de vida da aplicação.

O Ansible é o mais fácil de implementar, pois não requer a instalação de agentes nas máquinas clientes, pois utiliza o SSH (Secure Shell) para enviar as configurações. É uma ferramenta de código aberto baseada em Python, mas a configuração é codificada em arquivos YAML (YAML Ain't Markup Language), o que reduz a curva de aprendizado.

A abordagem do Puppet para o provisionamento de infraestrutura consiste em descrever o estado final desejado do sistema. Uma infraestrutura baseada no Puppet geralmente consiste em um servidor mestre com agentes em cada cliente. O Puppet é baseado em Ruby, mas usa uma DSL semelhante ao JSON (JavaScript Object Notation) para representar as configurações da máquina.

A abordagem do Chef é mais sobre escrever uma receita em uma DSL puramente baseada em Ruby que detalha as etapas necessárias para provisionar o sistema. Você pode usar o Chef como uma ferramenta cliente-servidor ou uma instalação isolada no modo "chef-solo".

Por fim, ao selecionar a tecnologia de orquestração, como o Docker, no contexto do DevOps, é importante levar em consideração se a infraestrutura de produção é baseada em virtualização ou se há intenção de migrar para tecnologias de containerização. O Docker é uma ferramenta que permite a automação de código, abrangendo tarefas como criação, teste e entrega ou implantação de software. Ele é amplamente utilizado para realizar integração contínua e entrega contínua (CI/CD) no processo de desenvolvimento de software.

Operações

Embora os pesquisadores tenham se concentrado bastante na construção e implantação, o DevOps também afeta as operações de infraestrutura. Portanto, você precisa de ferramentas para manter a estabilidade e o desempenho da sua infraestrutura.

Ferramentas de registro (logging). O registro às vezes é considerado como substituto da depuração, e os programadores são aconselhados a depurar em vez de registrar para não deixar rastros em um aplicativo. Mas do ponto de vista do DevOps, os rastros são necessários para o gerenciamento de aplicativos. A regra é registrar tudo e determinar o nível do registro (por exemplo, fatal, erro, aviso, info, trace ou debug). Para aplicativos não muito complexos, você pode

usar as ferramentas padrão (em Java, é o pacote java.util.logging). Para projetos mais complexos, você pode usar estruturas (frameworks) como o Log4j ou a família de ferramentas Log4j para diferentes linguagens, embora as ferramentas mais conhecidas para essas tarefas sejam os sistemas de gerenciamento de logs.

A ferramenta Loggly oferece um serviço baseado em nuvem para coletar dados de registro de aplicativos, plataformas e servidores em tempo real usando padrões abertos como HTTP ou syslog. A instalação é simples e a criação de painéis personalizados de desempenho e DevOps.

Ferramentas de registro (logging).

O Loggly também fornece uma funcionalidade de pesquisa poderosa. Ele pode ser integrado ao New Relic, tornando-o eficaz para identificar e isolar problemas e encontrar soluções.

O Graylog2 é um analisador de registros de código aberto que permite armazenar e pesquisar erros de registro. Em sua interface da web fácil de usar, você pode criar painéis ad hoc e recursos fortes de alerta. Ele lida com uma ampla variedade de formatos de dados e pode ser personalizado com um número considerável de plug-ins.

Ferramentas de monitoramento.

As ferramentas de monitoramento permitem que as organizações identifiquem e resolvam problemas na infraestrutura de TI antes que eles afetem os processos de negócio críticos. Essas ferramentas monitoram aspectos do sistema, como carga da CPU, alocação de RAM, estatísticas de tráfego de rede, consumo de memória e disponibilidade de espaço livre em disco. Elas acompanham continuamente a saúde do sistema e alertam os administradores quando detectam problemas, para que possam tomar ações corretivas.

O Nagios é uma conhecida ferramenta de código aberto para monitorar infraestruturas de TI, como estações de usuários finais, serviços de TI e componentes ativos de rede. Ele fornece uma interface web de fácil navegação com um painel interativo que inclui uma visão geral de alto nível de hosts, serviços e dispositivos de rede. Ele fornece gráficos de tendência e planejamento de capacidade que permitem que as organizações planejem atualizações de infraestrutura. Os plug-ins resolvem algumas das limitações da ferramenta, como a falta de descoberta automática de dispositivos e a dificuldade de configuração da ferramenta.

O New Relic, baseado em software como serviço, oferece uma interface poderosa para aplicativos web e monitora aplicativos móveis nativos para iOS e Android. Você pode monitorar aplicativos em execução em ambientes híbridos, em nuvem ou locais. Os painéis personalizáveis permitem que você integre recursos adicionais de monitoramento. Você pode gerar relatórios personalizados e definir alarmes de limite para receber notificações de eventos específicos.

O Cacti é um sistema baseado na web com uma interface intuitiva e fácil de usar. Sua configuração baseada em modelos permite um alto nível de personalização. Você pode criar gráficos, que podem ser exibidos no modo de visualização em árvore, além dos modos de lista padrão e de visualização prévia.

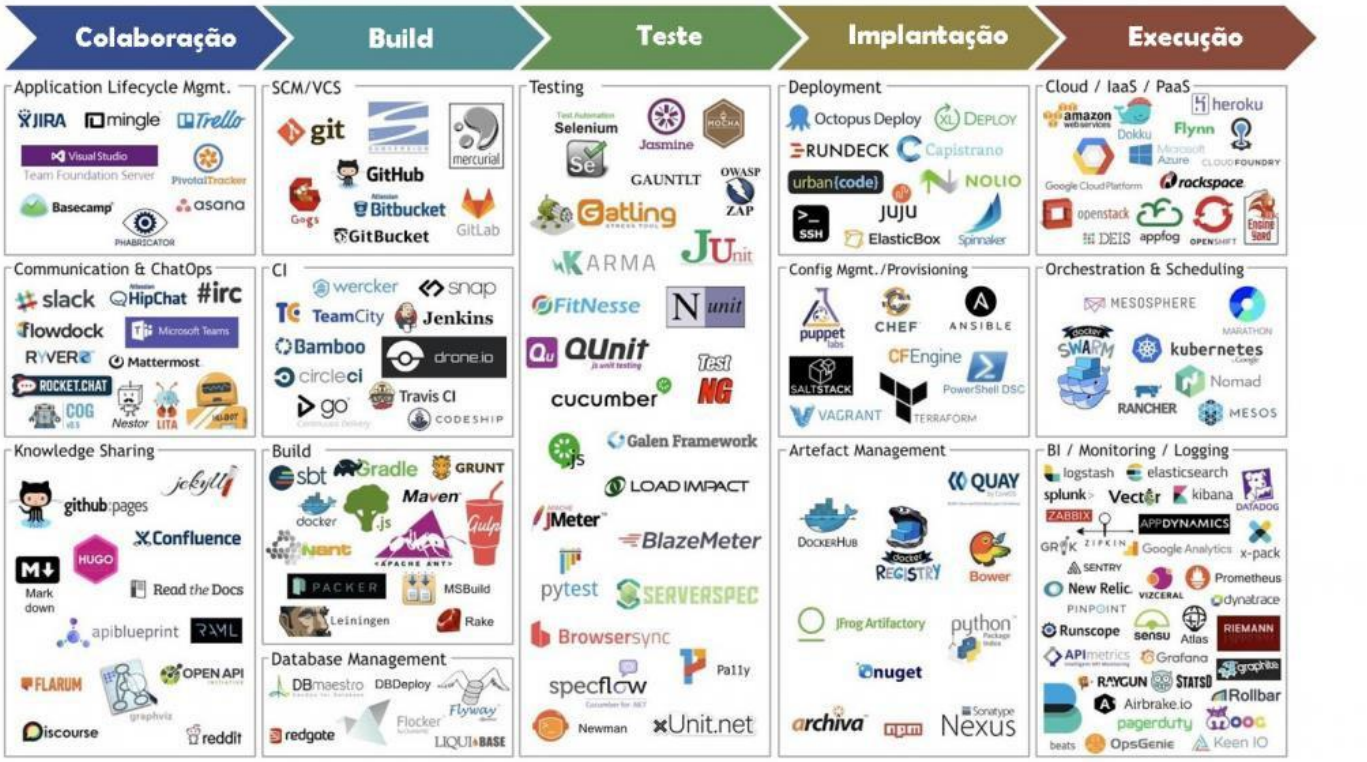
DevOps no Mundo Real

As indústrias de software e tecnologia da informação são voltadas para velocidade e eficiência. O DevOps surgiu como um paradigma para trazer produtos e recursos inovadores mais rapidamente ao mercado. Muitas tecnologias surgiram recentemente para suavizar a transição entre desenvolvimento e operações. Elas têm em comum práticas de entrega fluida, gerenciando assim a complexidade. Tecnologias para entrega rápida de aplicativos, como uma plataforma de busca ou venda, não se aplicam ao software embarcado. Mas certamente podemos aprender com essas abordagens e aplicá-las a outros domínios. Como mencionamos anteriormente, o surgimento de tecnologias de atualização rápida de software automotivo por meio de comunicação sem fio mostra que os princípios são transferíveis, levando em conta as rigorosas necessidades de segurança e continuidade funcional.

A nuvem e o desenvolvimento web têm sido adotantes precoces das práticas de DevOps e podem servir de guia para outros domínios. Por exemplo, a Amazon Web Services (AWS) oferece várias ferramentas para implementar a entrega contínua. Uma dessas ferramentas é o AWS Elastic Beanstalk, que suporta implantação contínua com uma abordagem fácil e, portanto, uma curva de aprendizado mais suave, embora com menos configurabilidade. O AWS OpsWorks oferece um caminho intermediário, no qual você pode codificar a infraestrutura; ele oferece integração com o Chef. Você também pode criar um modelo AWS CloudFormation, escrito em formato JSON, para provisionar infraestrutura de maneira repetível e controlar todos os componentes da infraestrutura na nuvem. Você pode usar o serviço AWS CodeDeploy para implantar aplicativos em várias máquinas virtuais (instâncias Elastic Compute Cloud) com um tempo de inatividade mínimo. Alternativamente, você pode usar o AWS CodePipeline. Esse serviço, lançado em 2015, integra compilação, teste e implementação.

Além disso, você pode usar outros componentes da AWS para oferecer suporte à entrega contínua. O AWS CodeCommit é um serviço de controle de origem gerenciado que hospeda repositórios privados. O AWS CloudWatch oferece uma infraestrutura de monitoramento e alerta que pode ajudar toda a equipe a trabalhar na confiabilidade e desempenho do aplicativo implantado.

Ao aproveitar essas ferramentas e práticas, as organizações podem adotar os princípios do DevOps e obter uma entrega de software mais rápida e eficiente em diversos domínios.



Testes

Após escrever o código, é muito importante verificar se tudo está funcionando corretamente. É aí que entram os testes de software. Eles são como um conjunto de verificações que garantem que o seu aplicativo está livre de erros e funciona como esperado.

Os testes são realizados para verificar se a aplicação possui erros e, caso sejam encontrados, corrigi-los. Os testes podem ser feitos de duas maneiras: manualmente, em que uma pessoa executa os testes de forma interativa, ou de forma automatizada, em que os testes são executados por meio de scripts ou ferramentas.

No contexto do DevOps, os testes integrados são fundamentais. Eles abrangem diferentes partes do sistema e têm o objetivo de garantir que todas elas funcionem corretamente em conjunto. Isso é importante porque, em um sistema complexo, diferentes componentes precisam interagir entre si, e os testes integrados garantem que isso aconteça sem problemas.

Para fazer esses testes, existem práticas como o Test-Driven Development (TDD) e o Behavior-Driven Development (BDD). Com o TDD, você escreve os testes antes mesmo de implementar o código. Isso ajuda a definir as expectativas e a criar um código mais confiável. Já o BDD se concentra em escrever testes de uma forma mais natural e legível, usando uma linguagem próxima ao negócio.

Uma vez que os testes integrados estão escritos, é possível automatizar sua execução. Isso significa que você pode criar um processo automatizado para executar os testes repetidamente, sempre que houver uma alteração no código. Isso é chamado de pipeline de integração contínua (CI). Quando o código é alterado, os testes são executados automaticamente, permitindo detectar rapidamente qualquer problema que possa ter surgido.

Além dos testes integrados, é importante também realizar outros tipos de testes. Os testes funcionais verificam se o aplicativo atende aos requisitos específicos de funcionalidade. Já os testes de integração garantem que as diferentes partes do sistema interajam corretamente entre si. Esses testes ajudam a verificar se o aplicativo funciona corretamente do começo ao fim, incluindo sua interação com outros componentes.

Quando todos esses tipos de testes estão integrados ao pipeline de CI, é possível ter um processo automatizado e contínuo de verificação da qualidade do aplicativo. Os testes são executados sempre que necessário, garantindo que o software seja testado de forma eficiente e identificando problemas o mais cedo possível. Existem também diferentes tipos de testes de software que podem ser realizados em diferentes estágios do desenvolvimento. Alguns desses testes incluem:

- Testes Unitários: São testes que verificam a funcionalidade de unidades individuais de código, como funções ou métodos. Eles são escritos pelos desenvolvedores e ajudam a garantir que cada parte do código esteja funcionando corretamente.

- Testes de Integração: São testes que verificam se as diferentes partes do sistema se integram corretamente entre si. Eles garantem que os componentes individuais funcionem em conjunto sem problemas.

- Testes de Sistema: São testes que verificam se a aplicação funciona corretamente como um todo, levando em consideração todos os seus componentes. Eles garantem que a aplicaçãoatenda aos requisitos estabelecidos.

- Testes de Aceitação: São testes que verificam se a aplicação está pronta para ser aceita pelo cliente ou usuário final. Eles se concentram em validar se a aplicação atende aos critérios de aceitação definidos.

Existem, ainda, diversas ferramentas disponíveis para auxiliar nos testes de software. Algumas das ferramentas mais usadas incluem:

- JUnit: O JUnit é um framework de teste de unidade para a linguagem de programação Java. Ele permite que os desenvolvedores escrevam e executem testes automatizados para verificar se cada unidade de código (métodos, classes, etc.) funciona corretamente. O JUnit é amplamente utilizado no desenvolvimento de software Java e é conhecido por sua simplicidade e eficiência.

- Karma: O Karma é uma ferramenta de execução de testes para aplicativos da web. Ele permite que os desenvolvedores executem testes em diferentes navegadores e plataformas, garantindo que o aplicativo seja compatível em todos os ambientes. O Karma é altamente configurável e suporta vários frameworks de teste, como o Jasmine.

- Jasmine: O Jasmine é um framework de teste de comportamento para JavaScript. Ele fornece uma sintaxe clara e concisa para escrever testes que verificam o comportamento esperado do código JavaScript. O Jasmine é amplamente utilizado no desenvolvimento de aplicativos web e é conhecido por sua legibilidade e facilidade de uso.

- TestNG: O TestNG é um framework de teste de unidade e integração para a linguagem de programação Java. Ele oferece recursos avançados, como a execução paralela de testes, relatórios detalhados e a capacidade de definir dependências entre testes. O TestNG é amplamente utilizado no desenvolvimento de software Java e é conhecido por sua flexibilidade e extensibilidade.

- Mocha: O Mocha é um framework de teste para JavaScript, tanto no navegador quanto no Node.js. Ele permite que os desenvolvedores escrevam testes simples e concisos usando uma sintaxe legível. O Mocha suporta a execução assíncrona de testes e oferece recursos como relatórios detalhados e geração de cobertura de código.

- Cucumber: O Cucumber é uma ferramenta de teste de comportamento que permite que os desenvolvedores escrevam testes em uma linguagem de fácil compreensão por pessoas não técnicas. Ele usa uma sintaxe baseada em linguagem natural chamada Gherkin, que descreve o comportamento do sistema em termos de cenários e passos de teste. O Cucumber é amplamente utilizado no desenvolvimento de software ágil e promove a colaboração entre desenvolvedores e partes interessadas não técnicas.

- Selenium: O Selenium é uma ferramenta de automação de testes para aplicativos da web. Ele permite que os desenvolvedores escrevam testes que interajam com o aplicativo como um usuário real, clicando em botões, preenchendo formulários e verificando resultados. O Selenium suporta vários navegadores e linguagens de programação e é amplamente utilizado no desenvolvimento de aplicativos web.

Gerenciamento de Configuração

O Gerenciamento de Configuração é um processo pelo qual você pode lidar com as alterações de forma sistemática. Esse processo garante que a integridade seja mantida o tempo todo e que o estado atual do sistema esteja em um estado conhecido e bom. As principais ferramentas utilizadas no gerenciamento de configuração são as seguintes:

- Ansible: O Ansible é uma ferramenta de TI de código aberto para gerenciar, automatizar, configurar servidores e, implantar aplicativos, a partir de uma localização central. Ele utiliza sua própria linguagem declarativa baseada em YAML para descrever as configurações do sistema e as tarefas que precisam ser executadas nos servidores. Essa linguagem de script é fácil de entender e permite a automação de tarefas em várias máquinas. Com o Ansible, você pode automatizar várias tarefas, como a configuração de servidores, provisionamento de infraestrutura, implantação de aplicativos e muito mais. Ele facilita o gerenciamento e a manutenção da infraestrutura, permitindo que você defina as configurações desejadas e asaplique de maneira consistente em diferentes ambientes.

- Terraform: O Terraform é uma ferramenta de infraestrutura como código que permite criar, modificar e versionar a infraestrutura de forma declarativa. É uma ferramenta de software de infraestrutura como código de código aberto criada pela HashiCorp. Os usuários definem e fornecem infraestrutura de data center usando uma linguagem de configuração declarativa conhecida como HashiCorp Configuration Language ou, opcionalmente, JSON. Ele oferece suporte a várias plataformas e provedores de nuvem, permitindo a criação e gerenciamento eficiente de recursos de infraestrutura.

- Puppet: O Puppet é uma ferramenta de automação de TI que simplifica a administração de configurações e o gerenciamento de infraestrutura. É um utilitário para gerenciamento de configuração de código livre. Ele roda em muitos sistemas Unix compatíveis, bem como em Microsoft Windows; e inclui sua própria linguagem declarativa para descrever a configuração do sistema. Ele permite definir e gerenciar a configuração do sistema de maneira centralizada, garantindo a consistência e o controle da infraestrutura.

- Rudder: O Rudder é uma ferramenta de automação de TI que facilita o gerenciamento e a configuração de vários servidores. É um utilitário de gerenciamento de configuração e auditoria de código aberto para ajudar a automatizar a configuração do sistema em grandes infraestruturas de TI. O Rudder conta com um agente local leve instalado em cada máquina gerenciada. Ele oferece recursos avançados de controle de conformidade e auditoria, permitindo a implantação e manutenção eficiente de configurações em escala.

- Salt: O Salt é uma ferramenta de gerenciamento de configuração e automação que permite gerenciar e provisionar sistemas de forma eficiente. Ele oferece recursos avançados de automação e escalabilidade, tornando-o adequado para ambientes complexos e distribuídos.

Implantação

Depois que sua aplicação foi testada e está pronta para ser incorporada à produção, o próximo estágio é a implantação. Aqui, a aplicação é implantada no ambiente de produção usando

diferentes ferramentas, dependendo do projeto ou da estrutura da aplicação. As principais ferramentas utilizadas para a etapa de implantação são as seguintes:

- O AWS CodeDeploy é um serviço fornecido pela Amazon Web Services (AWS) que permite automatizar as implantações de software em diferentes ambientes, como servidores da AWS, serviços de contêineres e servidores locais. Com o CodeDeploy, é possível realizar implantações de software de forma automatizada, eliminando a necessidade de processos manuais suscetíveis a erros. Isso agiliza o processo de implantação e ajuda a garantir a consistência e confiabilidade das implantações em diferentes ambientes de computação. O CodeDeploy é um serviço totalmente gerenciado pela AWS, o que significa que a infraestrutura subjacente e a escalabilidade são cuidadas pela AWS, permitindo que as equipes se concentrem no desenvolvimento e na entrega de software de forma mais eficiente.

- GoCD é uma ferramenta de construção e lançamento de código aberto da Thoughtworks. Ela é usada no desenvolvimento de software para automatizar a entrega contínua de software. Com o GoCD, equipes e organizações podem automatizar todo o processo de construção, teste e lançamento de software, desde o momento em que o código é enviado até a implantação. A ferramenta oferece suporte a infraestrutura moderna e ajuda empresas a obter software de forma mais eficiente.

- Octopus é um pacote de software utilizado para realizar cálculos de teoria funcional de densidade (DFT) de Kohn-Sham e teoria funcional de densidade dependente do tempo (TDDFT). Ele emprega pseudopotenciais e grades numéricas de espaço real para simular sistemas unidimensionais, bidimensionais e tridimensionais. Com o Octopus, é possível calcular propriedades como polarizabilidades, suscetibilidades magnéticas, espectros de absorção e realizar simulações de dinâmica molecular usando os métodos Ehrenfest e Car-Parrinello.

Containers

Containers são um conceito que surgiu no mercado atual para construir aplicações. A containerização permitiu aos usuários construir a aplicação com a ajuda de microservices, em que todos os pacotes e bibliotecas necessários para o serviço são empacotados em um único contêiner. Algumas das ferramentas mais populares presentes no mercado atualmente são as seguintes:

- Docker: O Docker é uma plataforma de contêineres de software que utiliza virtualização de nível de sistema operacional para empacotar e distribuir aplicativos de forma isolada. Com o Docker, os aplicativos podem ser executados em um ambiente consistente e portátil em diferentes sistemas operacionais e ambientes de implantação. Ele oferece uma solução eficiente e escalável para a entrega de software em contêineres, permitindo que os

aplicativos sejam executados de maneira isolada e com seus próprios recursos e configurações.

- Rancher: O Rancher é uma plataforma de gerenciamento de contêineres que oferece recursos avançados para implantar, gerenciar e orquestrar contêineres em ambientes de produção. Ele fornece uma interface amigável para gerenciar clusters de contêineres, facilitando a implantação e a escalabilidade de aplicativos baseados em contêineres.

- Azure Kubernetes: O Azure Kubernetes Service (AKS) é um serviço de orquestração de contêineres oferecido pela Microsoft Azure. Ele permite implantar, gerenciar e dimensionar facilmente aplicativos baseados em contêineres usando a plataforma Kubernetes. O AKS oferece recursos de automação e monitoramento para facilitar a implantação e a operação de clusters de contêineres no ambiente do Azure.

- Kubernetes: O Kubernetes é uma plataforma de orquestração de contêineres de código aberto que automatiza a implantação, dimensionamento e gerenciamento de aplicativos em contêineres. Projetado pelo Google e mantido pela Cloud Native Computing Foundation, o Kubernetes permite a execução eficiente e resiliente de aplicativos baseados em contêineres, oferecendo recursos avançados como balanceamento de carga, autorecuperação e dimensionamento automático. Sua ampla adoção e diversas distribuições tornaram-no uma escolha popular para executar e gerenciar cargas de trabalho nativas da nuvem.

- AWS ECS: O Amazon Elastic Container Service (ECS) é um serviço de orquestração de contêineres totalmente gerenciado fornecido pela Amazon Web Services (AWS). Ele permite implantar, gerenciar e escalar aplicativos conteinerizados de forma eficiente, integrando-se perfeitamente ao ambiente da AWS. Com recursos avançados de segurança e a opção do Amazon ECS Anywhere, é possível executar workloads de contêineres tanto na nuvem quanto on-premises. O ECS oferece uma solução fácil de usar para executar aplicativos em contêineres, fornecendo recursos de implantação, dimensionamento e gerenciamento de clusters de contêineres de maneira eficiente. Sua integração com outros serviços da AWS simplifica ainda mais o desenvolvimento de aplicativos escaláveis e altamente disponíveis.

- Codefresh: O Codefresh é uma plataforma de CI/CD de próxima geração projetada para aplicativos modernos. Ele oferece automação desde o código até a nuvem, com builds ultrarrápidos e implantações Canary e Blue/Green GitOps. Equipes de DevOps de empresas como GoodRx, Monday.com e Deloitte dependem do Codefresh para construir e implantar seu software de maneira segura e escalável. O Codefresh é especialmente adequado para aplicativos baseados em contêineres, fornecendo ferramentas e integração com várias plataformas populares, facilitando o desenvolvimento e a implantação contínua desses aplicativos.

Orquestração de Lançamento

Conforme o nome sugere, a Orquestração de Lançamento é uma abordagem para automatizar, coordenar e gerenciar as etapas completas do pipeline de lançamento de software. Essas ferramentas ajudam a automatizar o seu pipeline de CI/CD e permitem que você aproveite ferramentas e práticas que você pode ter utilizado durante o desenvolvimento do seu software. Alguns dos softwares de orquestração de lançamento são os seguintes:

- O XebiaLabs XL Release é uma ferramenta poderosa de orquestração e automação de lançamentos que permite que as organizações gerenciem e controlem de forma eficaz o processo de implantação de software. Com o XL Release, as equipes podem coordenar e executar fluxos de trabalho complexos de lançamento, garantindo visibilidade, rastreabilidade e controle ao longo de todo o ciclo de vida do lançamento. Essa ferramenta simplifica o gerenciamento de lançamentos, fornecendo recursos necessários para realizar implantações de software eficientes e confiáveis.

- O UrbanCode Deploy é uma solução de implantação de aplicativos desenvolvida pela IBM que automatiza o processo de entrega contínua e implantação contínua (CI/CD) e oferece recursos robustos de visibilidade, rastreabilidade e auditoria. Essa ferramenta de orquestração permite a implantação eficiente de aplicativos em ambientes complexos, oferecendo recursos avançados de automação e gerenciamento de versões. Com o UrbanCode Deploy, é possível realizar implantações rápidas e confiáveis de aplicativos em diversas plataformas e ambientes.

- O Spinnaker é uma plataforma de entrega contínua de código aberto, desenvolvida pelo Google, Netflix e outras empresas. Ele é projetado para simplificar a implantação de aplicativos em diversos ambientes e provedores de nuvem, permitindo uma entrega de software rápida e confiável. O Spinnaker oferece recursos avançados, como orquestração de implantação em vários estágios, gerenciamento de versões e integração com várias ferramentas e serviços. Com suporte para Kubernetes, Google Cloud Platform, AWS, Microsoft Azure e Oracle Cloud, o Spinnaker é uma opção versátil para organizações que buscam automação e agilidade na entrega de software.

- AWS CodePipeline é um serviço gerenciado pela Amazon Web Services (AWS) que oferece entrega contínua automatizada. Ele permite a automação do fluxo de trabalho de entrega de software, desde a integração do código até a implantação, proporcionando atualizações rápidas e confiáveis de aplicativos e infraestruturas. O CodePipeline integra-se perfeitamente a outros serviços da AWS, como o AWS CodeCommit, AWS CodeBuild e AWS CodeDeploy, oferecendo um pipeline completo de entrega contínua. Com o AWS CodePipeline, as equipes podem acelerar o processo de entrega de software, aumentar a eficiência e garantir a qualidade das atualizações.

Cloud

Cloud é o método de armazenar ou acessar seus dados pela internet, em vez de utilizar o seu próprio disco rígido. Atualmente, tudo é movido para a nuvem, executado na nuvem, acessado a partir da nuvem ou armazenado na nuvem. A aplicação ou o software que você desenvolve pode ser implantado na nuvem. Existem muitos provedores de nuvem no mercado atual, mas abaixo estão alguns provedores de nuvem populares que você pode considerar usar.

- Azure: O Azure é uma plataforma de computação em nuvem oferecida pela Microsoft. Ele fornece serviços de infraestrutura, como máquinas virtuais, armazenamento e redes, além de uma ampla gama de serviços gerenciados para desenvolvimento de aplicativos, análise de dados, inteligência artificial e muito mais. O Azure é conhecido por sua escalabilidade, confiabilidade e integração perfeita com outras ferramentas e serviços da Microsoft.

- AWS: A Amazon Web Services (AWS) é uma plataforma de computação em nuvem líder no mercado, oferecendo uma ampla variedade de serviços para ajudar empresas a construir e implantar aplicativos escaláveis e confiáveis. A AWS fornece serviços de infraestrutura, armazenamento, banco de dados, análise, inteligência artificial e muito mais. Ela é conhecida por sua flexibilidade, segurança e ampla presença global.

- Google Cloud: O Google Cloud é a plataforma de computação em nuvem oferecida pelo Google. Ele oferece serviços de computação, armazenamento, banco de dados, análise de dados, inteligência artificial e muito mais. O Google Cloud é conhecido por sua escalabilidade, desempenho e inovação, além de fornecer uma ampla gama de ferramentas e serviços para ajudar empresas a desenvolver, implantar e gerenciar aplicativos em escala.

- Lambda: O AWS Lambda é um serviço de computação sem servidor fornecido pela Amazon Web Services. Ele permite que os desenvolvedores executem código sem se preocupar com a infraestrutura subjacente. Com o AWS Lambda, é possível executar funções em resposta a eventos, como o upload de um arquivo para o armazenamento, uma solicitação HTTP ou uma atualização de banco de dados. O Lambda é conhecido por sua escalabilidade automática, pagamento por uso e fácil integração com outros serviços da AWS.

- IBM Cloud: O IBM Cloud é uma plataforma de computação em nuvem fornecida pela IBM. Ela oferece uma ampla gama de serviços, incluindo computação, armazenamento, banco de dados, inteligência artificial, análise de dados e muito mais. O IBM Cloud é conhecido por sua segurança, conformidade e capacidade de integração com ferramentas e serviços populares. Ele oferece opções flexíveis de implantação, incluindo nuvem pública, privada e híbrida, para atender às necessidades específicas das empresas.

AIOps

Operações de Inteligência Artificial ou AIOps é um termo amplo para análise de big data, inteligência artificial e outras tecnologias ou estruturas de aprendizado de máquina. Isso é usado para analisar os dados de um aplicativo utilizando conceitos como Big Data e Aprendizado de Máquina. Alguns dos tools mais comuns utilizados no mercado atual para AIOps são os seguintes:

Analytics

Analytics é usado para analisar os dados capturados por um aplicativo. Esse conjunto de ferramentas é usado principalmente para analisar e gerar relatórios inteligentes. Existem muitas ferramentas usadas para analisar os dados, mas poucas são muito populares na indústria DevOps. São elas:

- ITRS é um fornecedor líder de software de monitoramento e análise em tempo real para infraestrutura de TI, ajudando organizações a monitorar e gerenciar seus sistemas de TI e garantir alta disponibilidade e desempenho. Moogsoft é uma plataforma de AIOps que utiliza inteligência artificial e aprendizado de máquina para detectar e resolver incidentes e problemas de TI em tempo real. Logstash é um pipeline de processamento de dados de código aberto que ingere, processa e transforma dados de log para armazenamento e análise centralizados. Prometheus é um conjunto de ferramentas de monitoramento e alerta de código aberto projetado para ambientes nativos em nuvem, fornecendo uma solução flexível e escalável para coletar e analisar métricas. Fluentd é um coletor de dados de código aberto que permite a agregação e o processamento de dados de log de várias fontes, possibilitando o registro e a análise centralizados. Essas ferramentas desempenham um papel fundamental nas operações de TI modernas, ajudando as organizações a monitorar, analisar e solucionar problemas em sua infraestrutura e aplicativos de forma proativa e eficiente.

- Moogsoft é uma plataforma de análise de dados e gerenciamento de incidentes baseada em inteligência artificial. Ele usa algoritmos avançados para analisar grandes volumes de dados de log e métricas em tempo real, identificando padrões e anomalias que podem indicar problemas ou incidentes. O Moogsoft ajuda as equipes de operações a identificar e responder rapidamente a problemas, minimizando o impacto nos serviços de TI.

- Logstash é uma ferramenta de ingestão e processamento de dados open source. Ela permite coletar, transformar e enriquecer dados de diferentes fontes em tempo real, preparando-os para análise e armazenamento posterior. O Logstash suporta várias fontes de dados, como logs de aplicativos, eventos de rede e bancos de dados, e permite a aplicação de filtros e transformações para extrair informações relevantes.

- Prometheus é um sistema de monitoramento de código aberto projetado para coletar e armazenar métricas de tempo real. Ele foi desenvolvido para ser altamente escalável e oferecer suporte a ambientes distribuídos. O Prometheus coleta métricas de diferentes fontes, como aplicativos, sistemas e serviços, permitindo a análise e visualização desses dados. Ele também possui recursos avançados, como alertas e gráficos interativos.

- Fluentd é um coletor de registros e sistema de processamento de dados. Ele permite coletar, filtrar e encaminhar registros de várias fontes para diferentes destinos, como bancos de dados, sistemas de armazenamento ou serviços de análise. O Fluentd suporta uma ampla variedade de fontes e destinos, permitindo a integração fácil com outras ferramentas e sistemas. Ele oferece flexibilidade na configuração de pipelines de dados e é conhecido por sua escalabilidade e desempenho.

Monitoramento

Quando a aplicação é lançada na produção, é essencial monitorar a aplicação para garantir que seu desempenho seja adequado, que ela seja carregada rapidamente, que todos os recursos e funcionalidades da aplicação estejam funcionando corretamente, entre outros fatores. Portanto, para monitorar continuamente as aplicações, você pode usar as seguintes ferramentas:

- Nagios é uma ferramenta de monitoramento de código aberto amplamente utilizada para monitorar a infraestrutura de TI e os serviços de rede. Ele verifica regularmente os componentes do sistema, como servidores, roteadores, switches e aplicativos, em busca de problemas de disponibilidade e desempenho. O Nagios oferece recursos como alertas em tempo real, geração de relatórios e capacidade de escalonamento para atender às necessidades de monitoramento de grandes ambientes.

- Zabbix é uma plataforma de monitoramento de código aberto que permite monitorar a disponibilidade e o desempenho de vários componentes de infraestrutura, como servidores, redes e aplicativos. Ele usa uma arquitetura distribuída para coletar dados de monitoramento em tempo real e oferece recursos avançados, como detecção de anomalias, visualizações personalizadas e geração de relatórios detalhados. O Zabbix também oferece suporte a alertas e integração com outras ferramentas de gerenciamento de TI.

- Zenoss é uma plataforma de gerenciamento de operações de TI que inclui recursos de monitoramento, descoberta automática de ativos, gerenciamento de eventos e análise de desempenho. Ele permite monitorar vários componentes de infraestrutura, como servidores, redes, dispositivos de armazenamento e aplicativos, e fornece uma visão unificada e centralizada do ambiente de TI. O Zenoss também oferece recursos avançados, como análise preditiva, correlação de eventos e automação de tarefas de gerenciamento.

Segurança

Com o aumento do número de ameaças ou vulnerabilidades, garantir a segurança do aplicativo é um dos fatores mais importantes. Existem várias estratégias e tecnologias que você pode usar para proteger seu aplicativo contra diferentes tipos de ataques. No entanto, as principais ferramentas que você pode usar para garantir a segurança do seu aplicativo são as seguintes:

- SonarQube é uma plataforma de análise estática de código-fonte projetada para ajudar no gerenciamento da qualidade do código. Ele fornece um conjunto abrangente de ferramentas e recursos para analisar o código-fonte em várias linguagens de programação e identificar problemas de qualidade, como violações de boas práticas, bugs e vulnerabilidades de segurança. O SonarQube permite aos desenvolvedores detectar e corrigir problemas de código em um estágio inicial do ciclo de desenvolvimento, ajudando a melhorar a manutenção, a segurança e a eficiência dos aplicativos.

- Snort é um sistema de detecção de intrusões baseado em rede que monitora e analisa o tráfego em uma rede para identificar atividades maliciosas. Ele utiliza regras pré-configuradas para detectar e alertar sobre ameaças conhecidas, como ataques de negação de serviço, tentativas de invasão e atividades suspeitas. O Snort pode ser implantado em uma variedade de ambientes de rede e fornece recursos avançados, como análise de pacotes em tempo real, registro de eventos e integração com outros sistemas de segurança.

- Veracode é uma plataforma de segurança de aplicativos em nuvem que oferece análise estática e dinâmica de código para identificar e corrigir vulnerabilidades de segurança em aplicativos. Ele suporta várias linguagens de programação e fornece uma ampla gama de recursos de análise, como detecção de vulnerabilidades, análise de configuração de segurança, verificação de código malicioso e testes de penetração. O Veracode ajuda as organizações a melhorar a segurança de seus aplicativos, identificando e corrigindo falhas de segurança antes que elas sejam exploradas por atacantes.

Colaboração multidisciplinar

A colaboração multidisciplinar é uma característica fundamental da cultura DevOps. Ela envolve a quebra das barreiras tradicionais entre as equipes de desenvolvimento, operações e outras áreas envolvidas no ciclo de vida do software. Em vez de trabalhar de forma isolada, os profissionais dessas áreas colaboram de forma integrada, compartilhando conhecimentos, responsabilidades e objetivos comuns. Isso promove a comunicação eficiente, a troca de ideias e a resolução de problemas de forma ágil e colaborativa.

O teste manual no final do desenvolvimento é uma prática comum em muitos ambientes DevOps. Embora a automação de testes seja uma parte importante da abordagem DevOps, o teste manual também tem o seu lugar. Ele permite que os profissionais realizem testes exploratórios,

identifiquem possíveis problemas que podem ter passado despercebidos e validem a experiência do usuário final. O teste manual complementa os testes automatizados, proporcionando uma visão mais abrangente da qualidade do software.

A implantação contínua é outra característica marcante da cultura DevOps. Consiste em automatizar o processo de implantação de software, permitindo que as alterações sejam entregues ao ambiente de produção de forma rápida, frequente e confiável. Isso é alcançado por meio de práticas como integração contínua, entrega contínua e implantação contínua, nas quais as alterações são testadas e implantadas de maneira incremental e automatizada. A implantação contínua reduz os riscos, os gargalos e os atrasos associados às implantações tradicionais, permitindo que as equipes entreguem valor ao cliente de forma mais eficiente.

REFERÊNCIAS

Atlassian - Cinco princípios fundamentais de DevOps. Disponível em: https://www.atlassian.com/br/devops/what-is-devops.

Gene Kim, Jez Humble, John Willis, Patrick Debois. Manual de DevOps. Como Obter Agilidade, Confiabilidade e Segurança em Organizações Tecnológicas. Alta Books, 2018.

Jez Humble, David Farley. Entrega Contínua: Como Entregar Software de Forma Rápida e Confiável. Bookman, 2014.

Steve Matyas, Andrew Glover, Paul Duvall. Continuous Integration: Improving Software Quality and Reducing Risk. Addison-Wesley, 2007.

Valente, Marco Tulio. Engenharia de Software Moderna. 2022.

DevOps Lifecycle. Disponível em: https://www.educba.com/devops-lifecycle/

Ferramentas do DevOps Disponível em: https://www.atlassian.com/br/devops/devops-tools

DevOps Periodic Table. Disponível em: https://www.linkedin.com/pulse/devops-periodic-table-anurag-mor-1c

DevOps Periodic Table https://www.linkedin.com/pulse/devops-periodic-table-anurag-mor-1c

QUESTÕES COMENTADAS

DevOps

1. (CESPE – AGER-Mato Grosso– 2023). No que diz respeito aos conceitos de criptografia, à assinatura digital, aos conceitos utilizados em sistemas operacionais e às noções de DevOps, julgue o item seguinte.

   Colaboração multidisciplinar, teste manual no final do desenvolvimento e implantação contínua são algumas das características marcantes da cultura DevOps.

Comentários:

A cultura DevOps enfatiza a colaboração entre as equipes de desenvolvimento, operações e testes. Essa colaboração é necessária para garantir que o software seja desenvolvido, implantado e operado de forma eficiente e eficaz. O teste manual no final do desenvolvimento é uma prática tradicional, mas não é uma prática recomendada pelo DevOps. O DevOps enfatiza a automação de testes, pois permite que os testes sejam executados com mais frequência e eficiência. A implantação contínua é uma prática fundamental do DevOps. Ela visa garantir que as mudanças no software sejam implantadas em produção de forma rápida e segura.

Gabarito: Errado

---

2. (CESPE – AGER - Mato Grosso– 2023) Assinale a opção que apresenta exemplo de ferramenta que permite realizar automação de código — incluindo a execução de tarefas relacionadas à criação, ao teste e à entrega ou implantação de software — e, assim, realizar, no DevOps, integração contínua e entrega contínua (CI/CD).

   a) jenkins
   b) maven
   c) ansible
   d) puppet
   e) docker

Comentários:

A opção correta que apresenta um exemplo de ferramenta que permite a automação de código, incluindo a execução de tarefas relacionadas à criação, teste e entrega de software para integração contínua e entrega contínua (CI/CD) no contexto do DevOps é a opção A) Jenkins.

O Jenkins é uma ferramenta de automação de código aberto amplamente utilizada no DevOps. Ele permite criar pipelines de CI/CD, onde você pode automatizar as etapas de compilação, teste e implantação do software. Com o Jenkins, você pode configurar o fluxo de trabalho do seu projeto, realizar testes automatizados e implantar seu software de forma contínua e confiável. Ele é altamente configurável e suporta integração com várias ferramentas e serviços, tornando-o uma opção popular para automação no contexto do DevOps.

Já conhece os demais itens? Quer fazer uma revisão? Então vamos lá! O Maven é uma ferramenta de automação de build. Ele é frequentemente usado em projetos Java para gerenciar dependências, compilar o código-fonte, empacotar e distribuir o software. Com o Maven, você pode definir um arquivo de configuração chamado "pom.xml" que descreve as informações do projeto, as dependências necessárias e as etapas de compilação e empacotamento. O Maven automatiza o processo de build do projeto, tornando mais fácil para os desenvolvedores gerenciarem suas dependências e criar uma estrutura consistente para o código.

O Ansible é uma ferramenta de automação que permite gerenciar e configurar sistemas e infraestrutura de forma eficiente. Ele utiliza uma linguagem simples baseada em YAML para descrever as configurações e tarefas que precisam ser executadas nos servidores. Com o Ansible, você pode automatizar várias tarefas, como a configuração de servidores, provisionamento de infraestrutura, implantação de aplicativos e muito mais. Ele facilita o gerenciamento e a manutenção da infraestrutura, permitindo que você defina as configurações desejadas e as aplique de maneira consistente em diferentes ambientes.

O Puppet é uma ferramenta de gerenciamento de configuração que ajuda a manter a consistência e a configuração correta dos servidores e outras infraestruturas de TI. Com o Puppet, você pode descrever as configurações desejadas em um formato chamado "manifests" e o Puppet se encarrega de aplicar essas configurações em todos os servidores definidos. Ele automatiza tarefas como a instalação de pacotes, configuração de arquivos de configuração e garantia de que os servidores estejam de acordo com as políticas e padrões estabelecidos. O Puppet facilita o gerenciamento e a manutenção da infraestrutura em larga escala.

O Docker é uma plataforma que permite empacotar um aplicativo e suas dependências em um contêiner isolado. O contêiner é uma unidade autossuficiente que contém tudo o que o aplicativo precisa para ser executado, como código, bibliotecas e configurações. Com o Docker, você pode criar contêineres consistentes que podem ser executados em qualquer máquina que suporte Docker, independentemente do ambiente subjacente. Isso facilita a implantação consistente e repetível de aplicativos em diferentes ambientes, desde o desenvolvimento até a produção. O Docker é amplamente usado no contexto do DevOps para facilitar a entrega contínua, pois permite que os aplicativos sejam implantados de maneira rápida, fácil e confiável, garantindo a consistência entre os ambientes de desenvolvimento, teste e produção.

Gabarito: Letra A

3. (FGV – PGM - Niterói – 2023). Um Time de Desenvolvimento de Software (TDS) segue um protocolo automatizado para gerar, testar e combinar pacotes de software gerados separadamente. Todo software combinado precisa passar por um processo que inclui uma requisição formal ao Time de Operações (TO) de um Centro de Dados para executar um conjunto de testes, com o intuito de verificar vulnerabilidades no software antes de entrar em produção. Considerando os conceitos de DevOps e DevSecOps, o TDS e o TO estão falhando no princípio:

a) automação;
b) colaboração ágil;
c) produção contínua;
d) integração contínua;
e) ação centrada no cliente.

Comentários:

O gabarito é a letra A - automação. A falha no princípio da automação ocorre porque o processo descrito não é totalmente automatizado. Embora o Time de Desenvolvimento de Software (TDS) siga um protocolo automatizado para gerar, testar e combinar pacotes de software, há uma etapa em que é necessário fazer uma requisição formal ao Time de Operações (TO) para executar um conjunto de testes. Essa requisição formal indica que há uma intervenção manual no processo, o que vai contra o princípio da automação do DevOps e do DevSecOps.

As demais opções estão incorretas porque não estão diretamente relacionadas à falha mencionada. Colaboração ágil (opção B): Embora a colaboração entre o Time de Desenvolvimento de Software (TDS) e o Time de Operações (TO) seja mencionada, não há indicação de uma falha específica nessa colaboração.
Produção contínua (opção C): O texto menciona que os pacotes de software são combinados separadamente e passam por testes antes de entrar em produção. Portanto, não há uma falha específica no princípio da produção contínua.
Integração contínua (opção D): Embora o texto mencione que os pacotes de software são combinados, não há informações suficientes para determinar se há uma falha no princípio da integração contínua.
Ação centrada no cliente (opção E): O texto não menciona diretamente a relação com o cliente, portanto não é possível afirmar que há uma falha no princípio da ação centrada no cliente.

Gabarito: Letra A

4. (CONSULPAM – ICTIM - RJ – 2023) Uma das carreiras em ascensão na área de tecnologia, é a de DevOps, responsável por acelerar a colocação da solução no mercado, manter a estabilidade e a confiabilidade do sistema, melhorar o tempo médio de recuperação, entre outras ações. Assinale a alternativa que descreve as palavras que formam o acrônimo DevOps.

a) Desenho e Operação.
b) Desenho e Otimização.
c) Desenvolvimento e Operação.
d) Desenvolvimento e Otimização.

Comentários:

O termo DevOps é um acrônimo formado pelas palavras "Desenvolvimento" e "Operação". Essas palavras representam as principais áreas envolvidas no trabalho de DevOps. O objetivo do DevOps é promover a integração e colaboração entre as equipes de desenvolvimento e operações de TI, buscando melhorar a eficiência, qualidade e velocidade no desenvolvimento e entrega de software. Assim, a alternativa correta é a letra C, "Desenvolvimento e Operação".

Gabarito: Letra C

5. (Instituto Consulplan – MPE-MG – 2023) Manifesto para o desenvolvimento ágil de software defende "indivíduos e interações acima de processos e ferramentas, software operacional acima de documentação completa, colaboração dos clientes acima de negociação contratual e respostas a mudanças acima de seguir um plano". (Pressman e Maxim, 2021. P. 37.)
Considerando o exposto, analise as afirmativas a seguir.
I. Os princípios do Scrum são empregados para orientar as atividades de desenvolvimento dentro de um processo que incorpora as seguintes atividades metodológicas: planejar; codificar; construir; testar; e, distribuir.
II. A Extreme Programming (programação extrema) envolve um conjunto de regras e práticas constantes no contexto de quatro atividades metodológicas: planejamento; projeto; codificação; e, testes.
III. O projeto XP segue rigorosamente o princípio KISS (keep it simple, stupid!).

IV. As reuniões de equipe para o Kanban são semelhantes àquelas realizadas na metodologia XP.
V. O DevOps combina desenvolvimento (development) e operações (operations) e seu fluxo de trabalho envolve diversas etapas que formam ciclos contínuos até que o produto desejado exista de fato.

Está correto o que se afirma apenas em

a) I, II e V.
b) I, III e IV.
c) II, III e V.
d) III, IV e V.

Comentários:

Analisando as afirmativas: I. Os princípios do Scrum são empregados para orientar as atividades de desenvolvimento dentro de um processo que incorpora as seguintes atividades metodológicas: planejar; codificar; construir; testar; e, distribuir. Essa afirmativa está incorreta. Embora o Scrum seja um framework ágil amplamente utilizado no desenvolvimento de software, as atividades metodológicas mencionadas não são exclusivas do Scrum. O Scrum é baseado em iterações chamadas de sprints, nas quais o trabalho é planejado, desenvolvido, testado e entregue de forma iterativa e incremental.

II. A Extreme Programming (programação extrema) envolve um conjunto de regras e práticas constantes no contexto de quatro atividades metodológicas: planejamento; projeto; codificação; e, testes. Essa afirmativa está correta. A Extreme Programming (XP) é uma metodologia ágil que inclui atividades de planejamento, projeto, codificação e testes como parte de suas práticas. O XP enfatiza a colaboração entre a equipe, o feedback contínuo e a entrega rápida de software funcional.

III. O projeto XP segue rigorosamente o princípio KISS (keep it simple, stupid!). Essa afirmativa está correta. O projeto XP segue rigorosamente o princípio KISS, que enfatiza a importância de manter as coisas simples e evitar a complexidade desnecessária. O XP incentiva práticas de desenvolvimento simples e diretas para facilitar a compreensão e a manutenção do software.

IV. As reuniões de equipe para o Kanban são semelhantes àquelas realizadas na metodologia XP. Essa afirmativa está incorreta. Embora o Kanban e o XP sejam abordagens

ágeis, eles diferem em suas práticas e ênfases. As reuniões de equipe para o Kanban são focadas principalmente na visualização e no gerenciamento do fluxo de trabalho, enquanto o XP tem suas próprias reuniões específicas, como o planejamento do jogo (planning game), as reuniões de revisão e as reuniões de retrospectiva.

V. O DevOps combina desenvolvimento (development) e operações (operations) e seu fluxo de trabalho envolve diversas etapas que formam ciclos contínuos até que o produto desejado exista de fato. Essa afirmativa está correta. O DevOps combina desenvolvimento e operações, buscando integrar as equipes e os processos para acelerar a entrega de software. O fluxo de trabalho do DevOps envolve várias etapas que formam ciclos contínuos, incluindo desenvolvimento, testes, implantação e monitoramento, visando a entrega rápida e confiável do produto final. Portanto, a resposta correta é a alternativa c) II, III e V.

Gabarito: Letra C

---

6. (FUNDATEC – PROCERGS– 2023) O DevOps (desenvolvimento + operação) preza o estreitamento entre as áreas de desenvolvimento e infraestrutura através de ferramentas e metodologias, de modo que seja possível automatizar, monitorar, observar, testar e metrificar todas as etapas de desenvolvimento de software. Dentro dos processos de DevOps, que visam o aumento dessa qualidade e também a facilitação de colocar um projeto em produção, há um que é uma prática em que os times de desenvolvimento lançam novas funcionalidades de forma constante e automatizada. Quando uma nova funcionalidade é finalizada, automaticamente ela será disponibilizada no ambiente de testes e, posteriormente, no ambiente de produção e, em alguns casos, pode ir direto para produção. Assinale a alternativa que cita essa prática.

a) Design Patterns ou Padrões de Projeto.
b) Test Driven Development ou Desenvolvimento Orientado a Testes.
c) Continuous Delivery ou Entrega Contínua.
d) Behavior Driven Development ou Desenvolvimento Orientado ao Comportamento.
e) Continuous Integration ou Integração Contínua.

Comentários:

No contexto do DevOps, uma prática importante é a Entrega Contínua (Continuous Delivery). Essa prática consiste em lançar novas funcionalidades de forma constante e automatizada. Quando uma equipe de desenvolvimento finaliza uma nova funcionalidade, ela é automaticamente disponibilizada no ambiente de testes. Nesse ambiente, a funcionalidade passa por testes para garantir sua qualidade e funcionamento correto. Após ser aprovada nos testes, a funcionalidade pode ser promovida para o ambiente de produção, onde os usuários finais terão acesso a ela. A ideia por trás da Entrega Contínua é agilizar o processo de entrega de software, permitindo que as equipes entreguem novas funcionalidades de maneira rápida e eficiente. Em vez de lançar grandes atualizações em longos intervalos, a entrega contínua possibilita a disponibilização frequente de pequenas melhorias e recursos.

Gabarito: Letra C

---

7. (FUNDATEC – PROCERGS– 2023) No projeto em que você começará a trabalhar, você precisará de quatro engenheiros _____para programar o back-end (server) e front-end (client) da aplicação web. Além disso, precisará de mais um _____ para automatizar o deploy e a integração contínua da aplicação que será toda em AWS e GitHub Actions.

   Assinale a alternativa que preenche, correta e respectivamente, as lacunas do trecho acima.

   a) full-stack – engenheiro de dados
   b) full-stack – engenheiro DevOps
   c) de dados – analista de infraestrutura
   d) de dados – engenheiro DevOps
   e) full-stack – analisa de infraestrutura

Comentários:

No projeto em que você vai começar a trabalhar, serão necessários quatro engenheiros full-stack. Esses engenheiros serão responsáveis por programar tanto o back-end (parte do servidor) quanto o front-end (parte do cliente) da aplicação web. Eles devem ter habilidades para desenvolver todas as partes da aplicação.

Além dos engenheiros full-stack, será necessário mais um profissional: um engenheiro DevOps. Esse engenheiro terá a função de automatizar o processo de deploy (implantação) e a integração contínua da aplicação. Em outras palavras, ele será responsável por criar uma infraestrutura automatizada para que a aplicação seja implantada facilmente e de forma contínua. Ele utilizará ferramentas e tecnologias como AWS (Amazon Web Services) e GitHub Actions para realizar essa automação.

O engenheiro DevOps desempenha um papel importante na equipe, pois sua atuação permite que a equipe de desenvolvimento entregue o software de maneira mais eficiente, garantindo uma integração contínua suave e um ambiente de implantação estável.

Dessa forma, a resposta correta é a alternativa b) full-stack - engenheiro DevOps. Serão necessários quatro engenheiros full-stack para programar o back-end e front-end, e mais um engenheiro DevOps para automatizar o deploy e a integração contínua da aplicação, utilizando ferramentas como AWS e GitHub Actions.

Gabarito: Letra B

---

8. (CESPE – BANRISUL – 2022). Julgue o item que se segue, acerca de DevOps.

> Os processos envolvidos no DevOps são denominados, respectivamente, de planejar, construir, testar, codificar, operar, avaliar e relatar.

Comentários:

Pessoal, está errada a questão! Os processos envolvidos no DevOps são denominados, respectivamente, de planejar, desenvolver, construir, testar (DEV)(em inglês: Plan, code, build, test), Release/ lançamento ou liberação (início da fase ops) implantar, operar, monitorar (em inglês: release, deploy, operate, monitor) (OPS). Esses processos representam as etapas do ciclo de vida do desenvolvimento e operação de software no contexto do DevOps.

Gabarito: Errado

---

9. (CESPE – BNB – 2022). Considerando a figura a seguir, julgue o próximo item, acerca dosconceitos de DevOps.

A entrega contínua (CD) no DevOps é o processo de automatização que inclui a configuração e implantação de um aplicativo em um pipeline de produção, mas não abrange a compilação e o teste.

Comentários:

Errado. A entrega contínua (Continuous Delivery - CD) no DevOps abrange não apenas a configuração e implantação de um aplicativo, mas também a compilação e o teste automatizados. O objetivo da entrega contínua é garantir que o software seja entregue de maneira rápida, confiável e consistente, eliminando o trabalho manual e reduzindo o risco de erros humanos.

No contexto do DevOps, a entrega contínua envolve a automatização de todo o processo de desenvolvimento de software, desde o momento em que o código é escrito até o momento em que é implantado em produção. Isso inclui a compilação automatizada do código-fonte, a execução de testes automatizados para garantir a qualidade do software, a criação de artefatos de implantação e a implantação automatizada em ambientes de teste, homologação e produção.

A entrega contínua no DevOps é baseada na ideia de pipelines de entrega, que são fluxos de trabalho automatizados que movem o software através das diferentes etapas do processo de desenvolvimento e implantação. Esses pipelines são configurados para executar automaticamente todas as etapas necessárias, como compilação, teste, empacotamento e implantação, sem intervenção manual.

Gabarito: Errado

---

10. (CESPE – Petrobras – 2022). Julgue o item subsecutivo, relativos a DevOps e notação BPMN.

No DevOps, a integração contínua possui como uma de suas atividades a realização de testes; a fim de se obter os benefícios esperados convém automatizar os testes para poder executá-los para cada alteração feita no repositório principal.

Comentários:

Na prática, muitas organizações utilizam sistemas de controle de versão, como o Git, para automatizar processos de DevOps. Esses sistemas permitem que equipes de desenvolvedores trabalhem em conjunto em um mesmo projeto, realizando alterações no código-fonte de forma simultânea. Com a automação, é possível evitar conflitos e preservar o histórico do código.

Um exemplo dessa automação ocorre na integração contínua. Sempre que são feitas alterações no repositório principal do código-fonte, os testes automatizados são executados. Isso garante que, a cada mudança realizada, os testes necessários sejam executados de forma rápida e eficiente.

Dessa forma, a automação dos testes permite que a equipe de desenvolvimento mantenha a qualidade do software, identificando erros e falhas de forma ágil. Além disso, ao automatizar os testes, é possível ter um feedback constante sobre o estado do código, garantindo que ele permaneça funcional e livre de problemas. Essa prática contribui para a eficiência do processo de desenvolvimento, uma vez que os testes são executados de maneira automática e contínua, sem a necessidade de intervenção manual a cada alteração no código-fonte.

Gabarito: Correto

---

11. (CESPE – TRT - 8ª Região (PA e AP) – 2022). No contexto de DevOps e DevSecOps, o Proxy reverso
[+conteúdo]
a) permite que diferentes servidores e serviços apareçam como se fossem uma única unidade, ocultando servidores atrás do mesmo nome.
b) não permite o balanceamento de carga para distribuir o tráfego de entrada, uma vez que essa tarefa é realizada nativamente por um firewall.
c) é um servidor que reside na frente de um ou mais clientes, interceptando solicitações internas e externas de servidores web.
d) garante que os clientes se comuniquem diretamente com um servidor de origem na Web.
e) não permite criptografar e descriptografar comunicações SSL (ou TLS) para cada cliente.

Comentários:

No contexto de DevOps e DevSecOps, o proxy reverso é uma ferramenta importante que desempenha um papel fundamental na arquitetura de sistemas. Vamos entender melhor! O proxy reverso permite que diferentes servidores e serviços sejam vistos como uma única unidade. Isso significa que, do ponto de vista do cliente, todos esses servidores estão ocultos por trás de um único nome. Essa abstração simplifica o acesso aos serviços e facilita a sua gestão.

Diferentemente de um firewall, que nativamente não realiza o balanceamento de carga, o proxy reverso pode ser configurado para distribuir o tráfego de entrada entre os servidores de forma balanceada. Isso ajuda a otimizar o desempenho e garantir que cada servidor não seja sobrecarregado com uma carga excessiva de requisições. Na prática, o proxy reverso é um servidor posicionado à frente de um ou mais servidores web. Ele intercepta as solicitações dos clientes, tanto internos quanto externos, e as encaminha para o servidor correto. Dessa forma, o cliente se comunica diretamente com o proxy reverso, que então direciona a solicitação para o servidor de origem apropriado.

Além disso, o proxy reverso também pode ser configurado para criptografar e descriptografar as comunicações SSL (ou TLS) para cada cliente, proporcionando uma camada adicional de segurança na transmissão de dados.

Gabarito: Letra A

12. (CESPE – BANRISUL – 2022). Julgue o item a seguir, relativos à gestão de configuração DevOps e CI/CD.

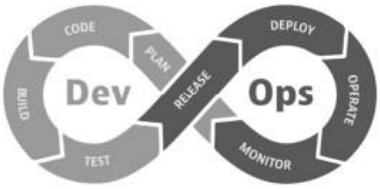
A integração contínua, a entrega contínua e a infraestrutura como código estão entre as melhores práticas de DevOps.

Comentários:

DevOps tem algumas práticas que ajudam na entrega de software de forma colaborativa e eficiente. Três delas são: integração contínua, entrega contínua e infraestrutura como código. A integração contínua é uma prática que envolve a integração frequente e automática de alterações no código de uma equipe de desenvolvimento. Isso significa que, sempre que um desenvolvedor faz uma alteração no código, essa alteração é integrada a um repositório compartilhado. Com a integração contínua, é possível identificar problemas de integração mais cedo, reduzindo o tempo necessário para detectar e corrigir erros. A entrega contínua é uma extensão da integração contínua. Ela envolve a automatização do processo de entrega de software para ambientes de teste ou produção. Com a entrega contínua, as alterações de código são testadas e disponibilizadas para os usuários finais de forma rápida e confiável. Isso permite um ciclo de feedback mais curto e uma resposta mais ágil às necessidades dos usuários. A infraestrutura como código (IaC) é uma prática que envolve a definição e gerenciamento da infraestrutura de um sistema de software por meio de código. Em vez de configurar manualmente servidores e recursos de infraestrutura, a infraestrutura é tratada como código, permitindo sua automação e reprodução consistente. O uso de ferramentas como o Git facilita a gestão e o versionamento do código de infraestrutura, permitindo que a infraestrutura seja tratada como uma parte integral do processo de desenvolvimento.

Gabarito: Correto

13. (CESPE – BNB– 2022). Considerando a figura a seguir, julgue o próximo item, acerca dos conceitos de DevOps.



Com base nas etapas do DevOps, é correto afirmar que a ferramenta Jenkins está mais relacionada à etapa monitor que à etapa deploy.

Comentários:

O Jenkins está mais relacionado à etapa de integração contínua (CI) do DevOps, em que ocorre a compilação, teste e integração do código. Ele é utilizado para automatizar tarefas como compilação, teste e criação de artefatos de software. O Jenkins é uma ferramenta altamente personalizável e flexível, sendo amplamente utilizado devido à sua extensibilidade e suporte a plugins e integrações. Embora possa fornecer recursos para monitorar o processo de construção e entrega do software, seu principal foco está na etapa de integração contínua, não na etapa de monitoramento. Portanto, a afirmação de que o Jenkins está mais relacionado à etapa de monitoramento do que à etapa de deploy está incorreta.

Gabarito: Errado

---

14. (CESPE – TCE-RJ – 2022). Acerca de arquitetura de TI, DevOps e COBIT 2019, julgue o item subsequente.

Em DevOps, além das ferramentas do ambiente de desenvolvimento integrado, são relevantes as ferramentas para gerenciamento de controle de fontes, trabalho colaborativo e planejamento de projetos.

Comentários:

O item está correto. Em DevOps, além das ferramentas do ambiente de desenvolvimento integrado, como IDEs (Integrated Development Environments) e ferramentas de automação de testes, são relevantes as ferramentas para gerenciamento de controle de fontes (como sistemas de controle de versão), trabalho colaborativo (como plataformas de colaboração e comunicação) e planejamento de projetos (como ferramentas de gestão de projetos e rastreamento de problemas). Por exemplo, um sistema de controle de versão é uma ferramenta essencial para rastrear e gerenciar as alterações feitas no código ao longo do tempo, garantindo que as versões mais recentes estejam disponíveis para todos os membros da equipe. Plataformas de colaboração e comunicação também são importantes, pois permitem que os desenvolvedores compartilhem ideias, discutam problemas e trabalhem juntos de forma eficiente.

Além disso, ferramentas de gestão de projetos e rastreamento de problemas são úteis para organizar e acompanhar as tarefas do projeto, garantindo que o trabalho seja planejado e executado de maneira eficaz. Essas ferramentas desempenham um papel fundamental na promoção da colaboração entre os membros da equipe, no acompanhamento das mudanças no código, no gerenciamento de tarefas e no suporte às práticas ágeis e de integração contínua.

Gabarito: Correto

---

15. (CESPE – TRT - 8ª Região (PA e AP) – 2022). A respeito de testes automatizados, no contexto de DevOps e DevSecOps, assinale a opção correta.

a) Em um teste unitário, os métodos da classe sendo testada e suas dependências podem ter relação com recursos externos.
b) Os bugs são detectados no final do ciclo de desenvolvimento, o que pode aumentar o tempo na criação de novos produtos.
c) Os testes unitários são testes de caixa preta com cada função que compõe o software.
d) O TDD (Test Driven Development) eleva o nível dos testes unitários e tem como característica criar a classe de testes antes da classe de produção, de forma que os testes guiem o código a ser implementado.
e) Os testes de integração são caracterizados pela verificação de partes internas do sistema, que se inter-relacionam entre si, conforme definido pelos clientes.

Comentários:

No contexto de DevOps e DevSecOps, o Test Driven Development (TDD) é uma prática em que os testes unitários são criados antes mesmo da implementação do código de produção. Isso significa que os desenvolvedores escrevem os testes para orientar o desenvolvimento do código. O objetivo principal do TDD é melhorar a qualidade dos testes unitários, garantindo que eles cubram adequadamente o código.

Ao utilizar o TDD, os desenvolvedores primeiro escrevem os testes, especificando o comportamento desejado do sistema. Em seguida, eles implementam o código necessário para fazer esses testes passarem. Dessa forma, os testes atuam como um guia para a implementação do código.

Essa abordagem traz várias vantagens. Em primeiro lugar, ajuda a garantir que o código seja mais confiável, pois é desenvolvido com base em testes bem definidos. Além disso, o TDD contribui para a detecção precoce de problemas, uma vez que os testes são criados antes da implementação do código. Isso permite que os desenvolvedores identifiquem e corrijam possíveis erros mais cedo no processo de desenvolvimento.

Outro benefício importante é que o TDD facilita a manutenção e a evolução do código. Como os testes estão sempre presentes e são executados automaticamente, é mais seguro fazer alterações e adicionar novas funcionalidades, pois os testes garantem que as partes existentes do sistema não sejam afetadas negativamente.

Gabarito: Letra D

---

16. (CESPE – APEX Brasil – 2022). É(são) etapa(s) do DevOps

I Imersão.

II Ideação.

III Prototipação.

Assinale a opção correta.

a) Nenhum item está .
b) Apenas os itens I e II estão s.
c) Apenas os itens I e III estão s.
d) Apenas os itens II e III estão s.

Comentários:

A opção correta é a letra A: Nenhum item está correto.

As etapas mencionadas (Imersão, Ideação e Prototipação) não são etapas específicas do DevOps. Elas são mais comumente associadas a processos de desenvolvimento de produtos, como o Design Thinking, em que se busca entender as necessidades dos usuários, gerar ideias e criar protótipos para validar conceitos.

Por outro lado, o DevOps é focado na integração e colaboração entre equipes de desenvolvimento e operações de TI, com o objetivo de entregar software de forma contínua, eficiente e com qualidade. As principais etapas do DevOps incluem Integração Contínua, Entrega Contínua e Operação Contínua. Isso envolve práticas como automação de processos, monitoramento constante e feedback contínuo para garantir um ciclo de desenvolvimento ágil e efetivo.

Gabarito: Letra A

---

17. (CESPE – APEX Brasil– 2022). São tarefas do DevOps

o(a)I monitoramento (monitor).

II planejamento (plan).

III codificação (code).

Assinale a opção correta.

a) Apenas os itens I e II estão s.
b) Apenas os itens I e III estão s.
c) Apenas os itens II e III estão s.
d) Todos os itens estão s.

Comentários:

A opção correta é a letra D: Todos os itens estão corretos. No contexto do DevOps, todas as tarefas mencionadas - monitoramento, planejamento e codificação - são consideradas atividades importantes e fazem parte das responsabilidades do DevOps.

O monitoramento é fundamental para garantir a disponibilidade, desempenho e segurança dos sistemas em produção. Isso envolve a coleta de dados, análise de métricas e monitoramento contínuo para identificar problemas e tomar medidas corretivas.

O planejamento é necessário para definir objetivos, prioridades e estratégias de implantação do software. Isso inclui o planejamento de releases, a definição de cronogramas, a coordenação de recursos e a gestão de mudanças.

A codificação refere-se ao desenvolvimento e manutenção do código-fonte do software. Os profissionais de DevOps podem estar envolvidos na criação, integração e automação de pipelines de desenvolvimento, além de implementar práticas de integração contínua e entrega contínua.

Gabarito: Letra D

---

18. (CESPE – BNB– 2022). A respeito de DevOps, julgue o item subsequente.

A organização que investir em DevOps deve estar preparada para automatizar seus processos mediante a execução de scripts pré-definidos.

Comentários:

O item está correto. Quando uma organização decide adotar a abordagem do DevOps, é importante que esteja pronta para automatizar seus processos usando scripts pré-definidos.

Automatizar processos é uma parte fundamental do DevOps, pois permite que tarefas repetitivas e demoradas sejam executadas de forma rápida, consistente e confiável. Ao criar scripts pré-definidos, a organização pode especificar as etapas necessárias para realizar uma determinada atividade, como compilar e implantar software, configurar ambientes ou executar testes.

Esses scripts podem ser executados repetidamente, garantindo a consistência nos processos e reduzindo a possibilidade de erros humanos. Além disso, a automação proporciona maior agilidade, permitindo que a organização responda rapidamente a mudanças e demandas do mercado.

Ao investir em DevOps, é essencial que a organização esteja preparada para adotar a automação como parte integrante de seus processos. Isso significa que os colaboradores devem estar dispostos a aprender e usar ferramentas de automação, além de criar e manter os scripts necessários para automatizar as atividades relevantes.

Gabarito: Correto

---

19. (CESPE – BNB– 2022). Considerando a figura a seguir, julgue o próximo item, acerca dos conceitos de DevOps.

A ferramenta RedHat Ansible está mais relacionada à etapa deploy do que à etapa plan.

Comentários:

A ferramenta RedHat Ansible está mais relacionada à etapa de implantação (deploy) do que à etapa de planejamento (plan) no contexto do DevOps. O Ansible é uma ferramenta de automação de TI que permite realizar a implantação automatizada de aplicativos e configurações em diferentes ambientes de produção.

Com o Ansible, é possível criar playbooks que descrevem as tarefas e configurações necessárias para a implantação de sistemas. Esses playbooks podem ser executados para provisionar e configurar servidores, instalar pacotes, configurar redes, entre outras ações relacionadas à implantação de aplicativos.

Gabarito: Correto

---

20. (CESPE – BANRISUL – 2022). Julgue o item que se segue, acerca de DevOps.

O repositório de artefatos armazena artefatos de construção produzidos por integração contínua e os disponibiliza para implantação automatizada em ambientes de teste, preparação e produção.

Comentários:

O item está correto porque um repositório de artefatos desempenha um papel essencial no processo de integração contínua e implantação automatizada. Ele armazena os artefatos de construção, que são os resultados do processo de compilação, como pacotes de distribuição e arquivos WAR. Esses artefatos são produzidos pela integração contínua e são disponibilizados no repositório para serem implantados automaticamente em diferentes ambientes, como testes, preparação e produção. O repositório de artefatos centraliza o armazenamento desses artefatos, facilitando o acesso e a implantação em diferentes estágios do ciclo de vida do software.

Além disso, o repositório de artefatos oferece uma forma de rastrear e controlar as versões dos artefatos, garantindo que as versões corretas sejam implantadas nos ambientes apropriados. Ele também permite a automatização do processo de implantação, permitindo que as equipes de desenvolvimento e operações automatizem a implantação de artefatos em diferentes ambientes, reduzindo erros e garantindo consistência. Em resumo, o repositório de artefatos desempenha um papel fundamental na entrega contínua de software, armazenando e disponibilizando os artefatos de construção para implantação automatizada em ambientes de teste, preparação e produção.

Gabarito: Correto

---

21. (CESPE – APEX Brasil – 2022). No DevOps, as mudanças feitas pelo desenvolvedor na soluçãode software, nas quais são feitos testes contra erros para depois serem enviadas a um repositório de versionamento de códigos, como o GitHub, representam a etapa de

a) deploy.
b) entrega contínua.
c) integração contínua.
d) operação.

Comentários:

A letra B é o gabarito correto porque a entrega contínua (continuous delivery) representa a etapa em que as mudanças feitas pelo desenvolvedor são testadas automaticamente e disponibilizadas em um repositório de código, como o GitHub. A entrega contínua visa minimizar o esforço na implantação de novos códigos, garantindo que as mudanças sejam testadas e prontas para serem implantadas em um ambiente de produção.

A entrega contínua tem como objetivo principal automatizar o processo de implantação, evitando atrasos causados por procedimentos manuais e garantindo uma entrega rápida e eficiente das aplicações aos clientes. Com a entrega contínua, as mudanças podem ser liberadas automaticamente para produção, sem a necessidade de intervenção manual, o que reduz a carga de trabalho das equipes de operações e acelera o ciclo de desenvolvimento.

Gabarito: Letra B

---

22. (FGV – TRT - 16ª REGIÃO (MA) – 2022) Considerando o DevOps e suas boas práticas,analise os itens a seguir:

I. Testes integrados são uma parte importante do processo DevOps. Esses testes devem levar em consideração as práticas de Test-Driven Development e Behavior-Driven Development, dessa forma a execução automática desses testes pode ser integrada ao pipeline de CI. No entanto, é importante integrar outros tipos de testes, como testes funcionais ou testes de integração, que permitem que o aplicativo seja testado funcionalmente do início ao fim com os outros componentes do seu ecossistema.
II. Recomenda-se automatizar apenas as tarefas críticas que envolvam poucas atualizações na implementação e nos testes dos aplicativos nas infraestruturas. Essas tarefas devem ser automatizadas em scripts que podem ser facilmente integradas e executadas em pipelines de CI/CD.
III. A construção de pipelines de CI/CD envolvem a escolha de ferramentas de DevOps adequadas pelas equipes considerando a natureza da empresa. E necessário levar em conta aspectos financeiros, avaliar entre ferramentas de código aberto e gratuitas e as proprietárias, que são mais ricas em recursos e suporte, mas exigem um investimento significativo.

Está correto apenas o que se afirma em

a) I e III.
b) I.
c) I e II.
d) II e III.
e) II.

Comentários:

A afirmação I está correta ao mencionar que os testes integrados são uma parte importante do processo DevOps. Esses testes garantem que as diferentes partes do sistema

funcionem corretamente em conjunto, permitindo testar funcionalidades do início ao fim, com integração aos outros componentes do ecossistema. Além disso, a afirmação menciona as práticas de Test-Driven Development (TDD) e Behavior-Driven Development (BDD), que são abordagens de desenvolvimento de software que incentivam a escrita de testes antes mesmo da implementação do código. A execução automática desses testes pode ser integrada ao pipeline de integração contínua (CI), garantindo que os testes sejam executados de forma automatizada e contínua.

A afirmação II está incorreta ao mencionar que apenas as tarefas críticas com poucas atualizações na implementação e nos testes dos aplicativos devem ser automatizadas. Na prática DevOps, a automação é um princípio fundamental, e recomenda-se automatizar o máximo possível de tarefas, independentemente de serem críticas ou não. Isso inclui tarefas como a compilação do código, execução de testes, implantação em ambientes de teste ou produção, provisionamento de infraestrutura, entre outras. A automação dessas tarefas é importante para garantir eficiência, consistência e agilidade no processo de entrega contínua (CI/CD).

A afirmação III está correta ao mencionar que a construção de pipelines de CI/CD envolve a escolha de ferramentas adequadas pelas equipes, considerando a natureza da empresa. É importante levar em conta aspectos financeiros ao escolher entre ferramentas de código aberto e gratuitas, assim como ferramentas proprietárias que possuem recursos e suporte mais abrangentes, mas geralmente exigem um investimento significativo. A escolha das ferramentas certas é fundamental para construir um pipeline de CI/CD eficiente e adaptado às necessidades da organização. Portanto, nosso gabarito é a opção A: I e III.

Gabarito: Letra A

---

23. (FGV – CGU– 2022) A equipe de redes de um órgão público está trabalhando para auxiliar no cumprimento das metas da equipe de desenvolvimento de sistemas do mesmo órgão e vislumbrou a possibilidade de utilização de DevOps. Para tal, a equipe de redes indicou a contratação de uma API em uma nuvem. A API indicada permite que os desenvolvedores e os administradores dos sistemas interajam com a infraestrutura de modo programático e em escala, evitando a instalação e a configuração dos recursos manualmente todas as vezes que precisam recriar um ambiente de desenvolvimento. Para essa atividade, a equipe de desenvolvimento utilizou a prática DevOps de:

a) comunicação e colaboração;

b) integração contínua;
c) entrega contínua;
d) microsserviços;
e) infraestrutura como código.

Comentários:

No cenário descrito, a equipe de redes do órgão público está colaborando com a equipe de desenvolvimento de sistemas para implementar o DevOps, uma abordagem que visa melhorar o processo de criação e gerenciamento dos ambientes de desenvolvimento. Para isso, eles propuseram a utilização de uma API em nuvem, que permite aos desenvolvedores e administradores interagir com a infraestrutura de forma programática, evitando a instalação e configuração manual dos recursos a cada criação de ambiente.

A prática de DevOps que está sendo aplicada nesse caso é a infraestrutura como código. Essa abordagem consiste em definir e gerenciar a infraestrutura utilizando scripts e configurações em vez de realizar as tarefas manualmente. Dessa forma, os desenvolvedores e administradores podem escrever o código que descreve a infraestrutura desejada e utilizar ferramentas que automatizam a criação e modificação dos recursos conforme necessário.

A infraestrutura como código traz diversos benefícios, como automação, consistência e escalabilidade. Ao tratar a infraestrutura como um código versionado, é possível acompanhar as alterações, realizar testes e manter um histórico das configurações. Isso torna o processo de criação e gerenciamento dos ambientes de desenvolvimento mais eficiente, confiável e facilita a colaboração entre as equipes de desenvolvimento e operações.

Gabarito: Letra E

---

24. (FGV – Senado Federal – 2022) Você foi contratado para liderar uma equipe de DevOps. Um dos objetivos da sua liderança é aumentar a velocidade das entregas e a qualidade de novos recursos das aplicações utilizando o desenvolvimento orientado a testes.

Assinale a opção que apresenta a ordem que descreve o ciclo de desenvolvimento orientado a testes.

a) Refatorar - > Escrever um código funcional
b) Escrever um caso de teste -> Refatorar
c) Refatorar - > Escrever um código funcional - > Escrever um caso de teste
d) Escrever um caso de teste -> Escrever um código funcional -> Refatorar
e) Escrever um código funcional -> Escrever um caso de teste -> Refatorar

Comentários:

A opção correta que descreve a ordem do ciclo de desenvolvimento orientado a testes é: e) Escrever um código funcional -> Escrever um caso de teste -> Refatorar. Nesse ciclo, conhecido como "Red-Green-Refactor" (Vermelho-Verde-Refatorar), o processo se inicia escrevendo um código funcional que implemente uma funcionalidade ou um novo recurso. Em seguida, escreve-se um caso de teste automatizado que valida o comportamento esperado dessa funcionalidade. O ciclo de desenvolvimento orientado a testes (TDD - Test-Driven Development) é uma prática que busca aumentar a qualidade e a confiabilidade do código por meio de testes automatizados. A sequência do ciclo é:

- Escrever um caso de teste: O desenvolvedor começa escrevendo um caso de teste que descreve o comportamento desejado para uma funcionalidade específica. Esse caso de teste deve falhar inicialmente, pois a funcionalidade ainda não está implementada.

- Escrever um código funcional mínimo: Em seguida, o desenvolvedor escreve o código mínimo necessário para que o caso de teste passe. Nesse ponto, o teste estará em estado "verde", indicando que a funcionalidade está implementada corretamente.

- Refatorar o código: Após o teste estar verde, o desenvolvedor pode refatorar o código, fazendo melhorias estruturais, eliminando duplicações, otimizando o código, entre outras ações. É importante ressaltar que, durante a refatoração, os testes automatizados devem ser executados para garantir que as mudanças não impactem o comportamento esperado da aplicação.

Esse ciclo é repetido várias vezes, em pequenos incrementos, à medida que novas funcionalidades são adicionadas ou alterações são feitas no código existente. O objetivo é garantir que o código seja testado continuamente e que todos os testes passem antes de qualquer nova alteração ser feita.

Essa abordagem traz benefícios significativos, pois os testes automatizados fornecem uma base sólida para a confiabilidade do código, permitindo identificar e corrigir problemas com maior agilidade. Além disso, o desenvolvimento orientado a testes ajuda a melhorar o design do código, pois a escrita dos testes antes da implementação incentiva a pensar sobre a interface e o comportamento esperado da funcionalidade a ser desenvolvida. Isso resulta em um código mais modular, de fácil manutenção e menos propenso a erros.

Gabarito: Letra D

---

25. (FCC – PGE-MG– 2022) A transição de DevOps para DevSecOps requer a compreensão e utilização de técnicas e práticas específicas que podem garantir a segurança do software. Uma especialista em Engenharia de Software recomendou, dentre outras, as seguintes ferramentas e tecnologias para essa transição em uma empresa:
I. É usado para verificar o código sem realmente executá-lo. Este tipo de ferramenta ajuda a encontrar vulnerabilidades em potencial no código-fonte, evitando que ocorram várias vulnerabilidades do tipo zero-day. Common Weakness Enumeration (CWE) é uma das classificações de avisos mais comuns produzidos por estas ferramentas. CWE é uma lista oficial ou dicionário de pontos fracos de segurança comuns exploráveis por invasores para obter acesso não autorizado ao sistema.
II. Da mesma forma que as ferramentas que executam testes de caixa preta, estes analisadores dinâmicos podem identificar vulnerabilidades do programa, como injeções de SQL, estouros de buffer e similares.
III. Este tipo de ferramenta analisa o comportamento do aplicativo, implementando uma análise de segurança contínua, sendo uma das tecnologias de segurança usadas em tempo de execução.

Os itens I, II e III correspondem, correta e respectivamente, a

a) RASP (Runtime Application Self-Protection) – SAST (Static Application Security Testing) – IAST (Interactive Application Security Testing).
b) IAST (Interactive Application Security Testing) – SAST (Static Application Security Testing) – DAST (Dynamic Application Security Testing).
c) SAST (Static Application Security Testing) – IAST (Interactive Application Security Testing) – DAST (Dynamic Application Security Testing).

d) IAST (Interactive Application Security Testing) – SAST (Static Application Security Testing) – RASP (Runtime Application Self-Protection).
e) SAST (Static Application Security Testing) – DAST (Dynamic Application Security Testing) – RASP (Runtime Application Self-Protection).

Comentários:

A transição de DevOps para DevSecOps envolve a incorporação de práticas e ferramentas de segurança no processo de desenvolvimento de software. Para auxiliar nessa transição, a especialista em Engenharia de Software recomendou algumas ferramentas e tecnologias. Vamos explicar cada uma delas de maneira mais simples:

I. Ferramentas de SAST (Static Application Security Testing): Essas ferramentas verificam o código-fonte do software em busca de vulnerabilidades sem executá-lo. É como fazer uma análise minuciosa do código para encontrar possíveis falhas de segurança. Elas ajudam a identificar vulnerabilidades antes que elas possam ser exploradas por invasores. Um exemplo de classificação comum de avisos produzidos por essas ferramentas é a CWE (Common Weakness Enumeration), que lista pontos fracos de segurança conhecidos.

II. Analisadores dinâmicos: Essas ferramentas são capazes de identificar vulnerabilidades durante a execução do programa. Elas realizam testes em tempo real e podem encontrar problemas como injeções de SQL e estouros de buffer. É como ter uma pessoa observando o programa enquanto ele está sendo executado para detectar qualquer comportamento suspeito.

III. Ferramentas de RASP (Runtime Application Self-Protection): Essas ferramentas monitoram o comportamento do aplicativo em tempo de execução. Elas analisam continuamente o aplicativo em funcionamento para detectar possíveis vulnerabilidades. É como ter um guarda de segurança que observa o comportamento do programa e toma medidas para protegê-lo contra ameaças em tempo real.

Assim, a sequência correta para a transição para o DevSecOps é utilizar ferramentas de SAST para verificar o código-fonte, seguidas por ferramentas de DAST que analisam o programa em tempo de execução, e por fim, ferramentas de RASP que monitoram continuamente o comportamento do aplicativo para protegê-lo contra ameaças.

Gabarito: Letra E

26. (FCC – TRT - 17ª Região (ES)– 2022) Para construir um pipeline como código noJenkins, um analista utilizou um arquivo de texto simples conhecido como

a) Pipejenkins que especifica o encadeamento dos estágios do processo DevSecOps.
b) Scripted que especifica o enlace crítico entre as etapas primárias DevOps.
c) JenkinsFile que normalmente contém um código Groovy Domain Specific Language.
d) JenkinsGroovy que especifica o enlace crítico usando código DomainPline.
e) Declarative que especifica o enlace crítico usando código GroovyPline.

Comentários:

Ao construir um pipeline como código no Jenkins, utilizamos um arquivo de texto chamado JenkinsFile. Essa é a opção correta (letra C). O JenkinsFile é um arquivo especial que contém instruções escritas em uma linguagem de programação chamada Groovy Domain Specific Language (DSL). O Groovy é uma linguagem poderosa e flexível usada para escrever o código do JenkinsFile. Ele nos permite definir e configurar o pipeline de integração e entrega contínua (CI/CD) como código, em vez de usar a interface gráfica do Jenkins. O JenkinsFile nos permite definir o pipeline de forma declarativa, o que significa que descrevemos os estágios, as etapas e as condições de execução de forma clara e direta. Ele nos permite organizar e encadear os estágios do processo DevSecOps de maneira adequada, garantindo que cada etapa seja executada na ordem correta.

Gabarito: Letra C

---

27. (IBFC – MGS– 2022) Quanto aos conceitos básicos de DevOps, segundo a Amazon,analise as afirmativas abaixo, dê valores Verdadeiro (V) ou Falso (F).

( ) Lança-se versões de software em ciclos periódicos, lineares, mas longos.
( ) As equipes de desenvolvimento ficam separadas dos operadores de software.
( ) DevOps é a combinação de filosofias culturais, práticas e ferramentas.

Assinale a alternativa que apresenta a sequência correta de cima para baixo.

a) V - F - F
b) V - V - F
c) F - V - V

d) F - F - V

Comentários:

Analisando as afirmativas:

( ) Lança-se versões de software em ciclos periódicos, lineares, mas longos.
Essa afirmativa é Falsa. No DevOps, busca-se lançar versões de software de forma contínua e incremental, em ciclos curtos. O objetivo é ter entregas frequentes e rápidas, em vez de ciclos longos e lineares, para atender às necessidades dos usuários de maneira ágil.

( ) As equipes de desenvolvimento ficam separadas dos operadores de software.
Essa afirmativa é Falsa. No DevOps, a colaboração e a integração são essenciais. As equipes de desenvolvimento e operações trabalham juntas, compartilhando responsabilidades e conhecimento. A ideia é quebrar as barreiras entre as equipes para alcançar uma abordagem colaborativa e eficiente.

( ) DevOps é a combinação de filosofias culturais, práticas e ferramentas.
Essa afirmativa é Verdadeira. DevOps vai além de ser apenas uma prática ou conjunto de ferramentas. É uma abordagem que combina filosofias culturais, práticas e ferramentas. Envolve uma mentalidade de colaboração, automação, integração contínua e entrega rápida de software.

Portanto, a sequência correta é: d) F - F - V. No DevOps, não são lançadas versões de software em ciclos longos e lineares, as equipes de desenvolvimento e operações trabalham juntas, e DevOps é uma combinação de filosofias culturais, práticas e ferramentas.

Gabarito: Letra D

---

28. (FUNDEP (Gestão de Concursos) – UFJF – 2022) Considere a sequência de comandos executados com sucesso em um repositório git para implementação de uma nova funcionalidade.

```
$ git branch cadastro-funcionario
$ git checkout cadastro-funcionario
...
$ git commit -a -m “Implementação do cadastro de funcionários”
```

```
$ git checkout master && git merge cadastro-funcionario
$ git push
```

Em relação à cultura DevOps e ao controle de versão, assinale a alternativa correta.

a) A funcionalidade desenvolvida foi registrada, mas não enviada ao servidor.
b) Foi utilizada a técnica de feature branch para implementação da funcionalidade.
c) O comando de merge foi aplicado no branch cadastro-funcionario.
d) O comando de commit pode não ter registrado todasasmudanças realizadasnaimplementação.
e) O comando de push foi realizado no branch cadastro-funcionario.

Comentários:

Em relação à cultura DevOps e ao controle de versão, podemos afirmar o seguinte:

a) A funcionalidade desenvolvida foi registrada, mas não enviada ao servidor.
Essa afirmativa está incorreta. O comando git push é usado para enviar as alterações para o servidor remoto, incluindo o branch cadastro-funcionario. Portanto, a funcionalidade foi enviada ao servidor.

b) Foi utilizada a técnica de feature branch para implementação da funcionalidade.
Essa afirmativa está correta. O comando git branch cadastro-funcionario cria um novo branch chamado cadastro-funcionario. Esse é um exemplo de utilização de feature branch, que é uma prática comum em DevOps para desenvolver novas funcionalidades em um branch separado antes de mesclá-las ao branch principal.

c) O comando de merge foi aplicado no branch cadastro-funcionario.
Essa afirmativa está incorreta. O comando git checkout master && git merge cadastro-funcionario indica que o merge foi realizado no branch master, não no cadastro-funcionario. O git checkout master muda para o branch master, e o git merge cadastro-funcionario realiza o merge do branch cadastro-funcionario no branch master.

d) O comando de commit pode não ter registrado todas as mudanças realizadas na implementação.
Essa afirmativa está correta. O comando git commit -a -m “Implementação do cadastro de funcionários” registra as alterações feitas na implementação, mas é possível que nem todas as mudanças tenham sido incluídas. Isso ocorre porque o comando -a do git commit

somente registra alterações em arquivos que já estão sob controle de versão. Se houver novos arquivos ou alterações em arquivos não rastreados, eles não serão incluídos no commit.

e) O comando de push foi realizado no branch cadastro-funcionario.
Essa afirmativa está incorreta. O comando git push foi realizado após o merge no branch master, portanto, o push foi feito no branch master, não no cadastro-funcionario.

Portanto, a alternativa correta é: b) Foi utilizada a técnica de feature branch para implementação da funcionalidade.

Gabarito: Letra B

---

29. (FEPESE – FAPESC– 2022) Muitas organizações têm adotado práticas de DevOpsno desenvolvimento de software.

Assinale a alternativa correta em relação ao assunto.
a) Baseado no modelo waterfall, DevOps é ideal para auxiliar equipes a manter um ritmo com modelos de desenvolvimento e entrega acelerados, tais como integração contínua e entrega para produção contínua (CI/CD).
b) Integração Contínua (Continuous integration - CI) é uma prática de desenvolvimento de software na qual desenvolvedores regularmente atualizam o seu código (realiza um commit) em um repositório compartilhado.
c) O uso de DevOps tem como objetivo criar uma cultura de colaboração entre as equipes de desenvolvimento e de operações que permite aumentar o fluxo de trabalho completado, reduzindo a quantidade de deploys, ao mesmo tempo aumentando a estabilidade e robustez do ambiente de produção.
d) A integração contínua (Continuous Integration - CI) foca na disponibilização de blocos de código completos para um repositório em intervalos regulares de tempo. Estes blocos de Código devem sempre estar em condições de serem executados para serem testados ou colocados em produção.
e) No uso de microservices, como uma arquitetura monolítica, CD permite aos desenvolvedores serem responsáveis por partes maiores e gerenciáveis do código que implementam funcionalidades individuais e trabalhar nestas em paralelo.

Comentários:

A alternativa correta é a letra B. A integração contínua (Continuous Integration - CI) é uma prática no desenvolvimento de software em que os desenvolvedores atualizam regularmente o seu código em um repositório compartilhado, também conhecido como repositório de controle de versão. Essa prática tem como objetivo principal integrar as alterações de código feitas por diferentes desenvolvedores de forma frequente, para detectar problemas de integração o mais cedo possível.

Ao realizar um commit (atualização do código) no repositório compartilhado, o código passa por um processo de integração automática, em que são executados testes automatizados para verificar se o código integrado não causa problemas de compatibilidade ou erros no sistema. Essa abordagem permite identificar e resolver rapidamente os problemas de integração, além de promover uma maior colaboração entre os membros da equipe.

A integração contínua é uma prática fundamental no contexto do DevOps, pois possibilita a entrega rápida e frequente de novas funcionalidades, mantendo a estabilidade e qualidade do software. Com a integração contínua, as equipes podem trabalhar de forma mais ágil, diminuindo os riscos de conflitos e problemas de integração tardia.

Gabarito: Letra B

---

30. (IBADE – SEA-SC – 2022) Em um modelo DevOps existe um método para entregar aplicações com frequência aos clientes, visando integração, entrega e implantação contínuas. Chamamos esse método de:

a) git
b) pipeline
c) flowchart
d) reentrante
e) CI/CD

Comentários:

A alternativa correta é a letra e) CI/CD. No modelo DevOps, a prática de integração contínua (CI) está relacionada à integração frequente e automatizada do código fonte em um repositório compartilhado. Isso permite que as equipes trabalhem de forma colaborativa, integrando suas alterações de forma contínua e automatizada.

Já a prática de entrega contínua (CD) envolve a automatização do processo de implantação do software em ambientes de produção. Com a entrega contínua, as alterações de software podem ser entregues rapidamente aos usuários, reduzindo o tempo de espera e permitindo um feedback mais rápido.

A combinação dessas práticas, CI/CD, é essencial no DevOps, pois permite uma entrega mais ágil, confiável e frequente de software, com a integração e implantação contínuas, mantendo a qualidade e a estabilidade do sistema.

Gabarito: Letra E

---

31. (UFAC – UFAC– 2022) O administrador DevOps acessou um servidor Linux da empresa, ao utilizar o comando “df” no “i” apontou 100% de uso. Julgue a informação recebida pelo console:

a) A memória swap atingiu sua capacidade máxima
b) A memória RAM não possui espaço disponível
c) O número de conexões máximas foi atingido
d) O sistema de arquivos raiz está cheio
e) Nenhuma das alternativas está correta.

Comentários:

Ao utilizar o comando "df" com a opção "i" em um servidor Linux e obter 100% de uso, a informação recebida pelo console indica que o número de inodes disponíveis no sistema de arquivos está totalmente utilizado. Isso significa que não há mais espaço disponível para armazenar informações sobre novos arquivos e diretórios.

Portanto, a resposta correta é a opção d) O sistema de arquivos raiz está cheio. Isso significa que o diretório raiz do sistema, que geralmente é montado em "/", está com sua capacidade de armazenamento esgotada. Isso pode afetar o funcionamento do servidor, causar problemas de desempenho e impedir a criação de novos arquivos.

Gabarito: Letra D

QUESTÕES COMENTADAS

DevOps

1. (CESPE – AGER-Mato Grosso– 2023). No que diz respeito aos conceitos de criptografia, à assinatura digital, aos conceitos utilizados em sistemas operacionais e às noções de DevOps, julgue o item seguinte.

   Colaboração multidisciplinar, teste manual no final do desenvolvimento e implantação contínua são algumas das características marcantes da cultura DevOps.

2. (CESPE – AGER - Mato Grosso– 2023) Assinale a opção que apresenta exemplo de ferramenta que permite realizar automação de código — incluindo a execução de tarefas relacionadas à criação, ao teste e à entrega ou implantação de software — e, assim, realizar, no DevOps, integração contínua e entrega contínua (CI/CD).

   a) jenkins
   b) maven
   c) ansible
   d) puppet
   e) docker

3. (FGV – PGM - Niterói – 2023). Um Time de Desenvolvimento de Software (TDS) segue um protocolo automatizado para gerar, testar e combinar pacotes de software gerados separadamente. Todo software combinado precisa passar por um processo que inclui uma requisição formal ao Time de Operações (TO) de um Centro de Dados para executar um conjunto de testes, com o intuito de verificar vulnerabilidades no software antes de entrar em produção. Considerando os conceitos de DevOps e DevSecOps, o TDS e o TO estão falhando no princípio:

   a) automação;
   b) colaboração ágil;
   c) produção contínua;
   d) integração contínua;
   e) ação centrada no cliente.

4. (CONSULPAM – ICTIM - RJ – 2023) Uma das carreiras em ascensão na área de tecnologia, é a de DevOps, responsável por acelerar a colocação da solução no mercado, manter a estabilidade e a confiabilidade do sistema, melhorar o tempo médio

de recuperação, entre outras ações. Assinale a alternativa que descreve as palavras que formam o acrônimo DevOps.

a) Desenho e Operação.
b) Desenho e Otimização.
c) Desenvolvimento e Operação.
d) Desenvolvimento e Otimização.

5. (Instituto Consulplan – MPE-MG – 2023) Manifesto para o desenvolvimento ágil de software defende "indivíduos e interações acima de processos e ferramentas, software operacional acima de documentação completa, colaboração dos clientes acima de negociação contratual e respostas a mudanças acima de seguir um plano". (Pressman e Maxim, 2021. P. 37.)
Considerando o exposto, analise as afirmativas a seguir.
I. Os princípios do Scrum são empregados para orientar as atividades de desenvolvimento dentro de um processo que incorpora as seguintes atividades metodológicas: planejar; codificar; construir; testar; e, distribuir.
II. A Extreme Programming (programação extrema) envolve um conjunto de regras e práticas constantes no contexto de quatro atividades metodológicas: planejamento; projeto; codificação; e, testes.
III. O projeto XP segue rigorosamente o princípio KISS (keep it simple, stupid!).
IV. As reuniões de equipe para o Kanban são semelhantes àquelas realizadas na metodologia XP.
V. O DevOps combina desenvolvimento (development) e operações (operations) e seu fluxo de trabalho envolve diversas etapas que formam ciclos contínuos até que o produto desejado exista de fato.

Está correto o que se afirma apenas em

a) I, II e V.
b) I, III e IV.
c) II, III e V.
d) III, IV e V.

6. (FUNDATEC – PROCERGS– 2023) O DevOps (desenvolvimento + operação) preza o estreitamento entre as áreas de desenvolvimento e infraestrutura através de ferramentas e metodologias, de modo que seja possível automatizar, monitorar,

observar, testar e metrificar todas as etapas de desenvolvimento de software. Dentro dos processos de DevOps, que visam o aumento dessa qualidade e também a facilitação de colocar um projeto em produção, há um que é uma prática em que os times de desenvolvimento lançam novas funcionalidades de forma constante e automatizada. Quando uma nova funcionalidade é finalizada, automaticamente ela será disponibilizada no ambiente de testes e, posteriormente, no ambiente de produção e, em alguns casos, pode ir direto para produção. Assinale a alternativa que cita essa prática.

a) Design Patterns ou Padrões de Projeto.
b) Test Driven Development ou Desenvolvimento Orientado a Testes.
c) Continuous Delivery ou Entrega Contínua.
d) Behavior Driven Development ou Desenvolvimento Orientado ao Comportamento.
e) Continuous Integration ou Integração Contínua.

7. (FUNDATEC – PROCERGS– 2023) No projeto em que você começará a trabalhar, você precisará de quatro engenheiros _____ para programar o back-end (server) e front-end (client) da aplicação web. Além disso, precisará de mais um _____ para automatizar o deploy e a integração contínua da aplicação que será toda em AWS e GitHub Actions.

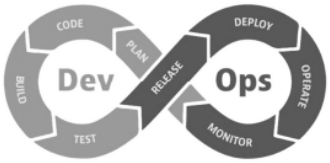
Assinale a alternativa que preenche, correta e respectivamente, as lacunas do trecho acima.

a) full-stack – engenheiro de dados
b) full-stack – engenheiro DevOps
c) de dados – analista de infraestrutura
d) de dados – engenheiro DevOps
e) full-stack – analisa de infraestrutura

8. (CESPE – BANRISUL – 2022). Julgue o item que se segue, acerca de DevOps.

Os processos envolvidos no DevOps são denominados, respectivamente, de planejar, construir, testar, codificar, operar, avaliar e relatar.

9. (CESPE – BNB – 2022). Considerando a figura a seguir, julgue o próximo item, acerca dos conceitos de DevOps.



A entrega contínua (CD) no DevOps é o processo de automatização que inclui a configuração e implantação de um aplicativo em um pipeline de produção, mas não abrange a compilação e o teste.

10. (CESPE – Petrobras – 2022). Julgue o item subsecutivo, relativos a DevOps e notação BPMN.

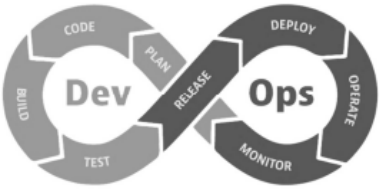
No DevOps, a integração contínua possui como uma de suas atividades a realização de testes; a fim de se obter os benefícios esperados convém automatizar os testes para poder executá-los para cada alteração feita no repositório principal.

11. (CESPE – TRT - 8ª Região (PA e AP) – 2022). No contexto de DevOps e DevSecOps, o Proxy reverso
[+conteúdo]
a) permite que diferentes servidores e serviços apareçam como se fossem uma única unidade, ocultando servidores atrás do mesmo nome.
b) não permite o balanceamento de carga para distribuir o tráfego de entrada, uma vez que essa tarefa é realizada nativamente por um firewall.
c) é um servidor que reside na frente de um ou mais clientes, interceptando solicitações internas e externas de servidores web.
d) garante que os clientes se comuniquem diretamente com um servidor de origem na Web.
e) não permite criptografar e descriptografar comunicações SSL (ou TLS) para cada cliente.

12. (CESPE – BANRISUL – 2022). Julgue o item a seguir, relativos à gestão de configuração DevOps e CI/CD.

A integração contínua, a entrega contínua e a infraestrutura como código estão entre as melhores práticas de DevOps.

13. (CESPE – BNB– 2022). Considerando a figura a seguir, julgue o próximo item, acerca dos conceitos de DevOps.



Com base nas etapas do DevOps, é correto afirmar que a ferramenta Jenkins está mais relacionada à etapa monitor que à etapa deploy.

14. (CESPE – TCE-RJ – 2022). Acerca de arquitetura de TI, DevOps e COBIT 2019, julgue o item subsequente.

Em DevOps, além das ferramentas do ambiente de desenvolvimento integrado, são relevantes as ferramentas para gerenciamento de controle de fontes, trabalho colaborativo e planejamento de projetos.

15. (CESPE – TRT - 8ª Região (PA e AP) – 2022). A respeito de testes automatizados, no contexto de DevOps e DevSecOps, assinale a opção correta.

a) Em um teste unitário, os métodos da classe sendo testada e suas dependências podem ter relação com recursos externos.
b) Os bugs são detectados no final do ciclo de desenvolvimento, o que pode aumentar o tempo na criação de novos produtos.
c) Os testes unitários são testes de caixa preta com cada função que compõe o software.
d) O TDD (Test Driven Development) eleva o nível dos testes unitários e tem como característica criar a classe de testes antes da classe de produção, de forma que os testes guiem o código a ser implementado.
e) Os testes de integração são caracterizados pela verificação de partes internas do sistema, que se inter-relacionam entre si, conforme definido pelos clientes.

16. (CESPE – APEX Brasil – 2022). É(são) etapa(s) do DevOps

I Imersão.

II Ideação.

III Prototipação.

Assinale a opção correta.

a) Nenhum item está .
b) Apenas os itens I e II estão s.
c) Apenas os itens I e III estão s.
d) Apenas os itens II e III estão s.

17. (CESPE – APEX Brasil– 2022). São tarefas do DevOps o(a)

I monitoramento (monitor).

II planejamento (plan).

III codificação (code).

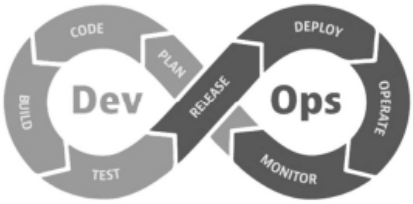
Assinale a opção correta.

a) Apenas os itens I e II estão s.
b) Apenas os itens I e III estão s.
c) Apenas os itens II e III estão s.
d) Todos os itens estão s.

18. (CESPE – BNB– 2022). A respeito de DevOps, julgue o item subsequente.

A organização que investir em DevOps deve estar preparada para automatizar seus processos mediante a execução de scripts pré-definidos.

19. (CESPE – BNB– 2022). Considerando a figura a seguir, julgue o próximo item, acerca dos conceitos de DevOps.



A ferramenta RedHat Ansible está mais relacionada à etapa deploy do que à etapa plan.

20. (CESPE – BANRISUL – 2022). Julgue o item que se segue, acerca de DevOps.

O repositório de artefatos armazena artefatos de construção produzidos por integração contínua e os disponibiliza para implantação automatizada em ambientes de teste, preparação e produção.

21. (CESPE – APEX Brasil – 2022). No DevOps, as mudanças feitas pelo desenvolvedor na solução de software, nas quais são feitos testes contra erros para depois serem enviadas a um repositório de versionamento de códigos, como o GitHub, representam a etapa de

a) deploy.
b) entrega contínua.

c) integração contínua.
d) operação.

22. (FGV – TRT - 16ª REGIÃO (MA) – 2022) Considerando o DevOps e suas boas práticas, analise os itens a seguir:

I. Testes integrados são uma parte importante do processo DevOps. Esses testes devem levar em consideração as práticas de Test-Driven Development e Behavior-Driven Development, dessa forma a execução automática desses testes pode ser integrada ao pipeline de CI. No entanto, é importante integrar outros tipos de testes, como testes funcionais ou testes de integração, que permitem que o aplicativo seja testado funcionalmente do início ao fim com os outros componentes do seu ecossistema.
II. Recomenda-se automatizar apenas as tarefas críticas que envolvam poucas atualizações na implementação e nos testes dos aplicativos nas infraestruturas. Essas tarefas devem ser automatizadas em scripts que podem ser facilmente integradas e executadas em pipelines de CI/CD.
III. A construção de pipelines de CI/CD envolvem a escolha de ferramentas de DevOps adequadas pelas equipes considerando a natureza da empresa. E necessário levar em conta aspectos financeiros, avaliar entre ferramentas de código aberto e gratuitas e as proprietárias, que são mais ricas em recursos e suporte, mas exigem um investimento significativo.

Está correto apenas o que se afirma em

a) I e III.
b) I.
c) I e II.
d) II e III.
e) II.

23. (FGV – CGU– 2022) A equipe de redes de um órgão público está trabalhando para auxiliar no cumprimento das metas da equipe de desenvolvimento de sistemas do mesmo órgão e vislumbrou a possibilidade de utilização de DevOps. Para tal, a equipe de redes indicou a contratação de uma API em uma nuvem. A API indicada permite que os desenvolvedores e os administradores dos sistemas interajam com a infraestrutura de modo programático e em escala, evitando a instalação e a configuração dos recursos manualmente todas as vezes que precisam recriar um ambiente de desenvolvimento. Para essa atividade, a equipe de desenvolvimento utilizou a prática DevOps de:

a) comunicação e colaboração;
b) integração contínua;
c) entrega contínua;
d) microsserviços;
e) infraestrutura como código.

24. (FGV – Senado Federal – 2022) Você foi contratado para liderar uma equipe de DevOps. Um dos objetivos da sua liderança é aumentar a velocidade das entregas e a qualidade de novos recursos das aplicações utilizando o desenvolvimento orientado a testes.

Assinale a opção que apresenta a ordem que descreve o ciclo de desenvolvimento orientado a testes.

a) Refatorar - > Escrever um código funcional
b) Escrever um caso de teste -> Refatorar
c) Refatorar - > Escrever um código funcional - > Escrever um caso de teste
d) Escrever um caso de teste -> Escrever um código funcional -> Refatorar
e) Escrever um código funcional -> Escrever um caso de teste -> Refatorar

25. (FCC – PGE-MG– 2022) A transição de DevOps para DevSecOps requer a compreensão e utilização de técnicas e práticas específicas que podem garantir a segurança do software. Uma especialista em Engenharia de Software recomendou, dentre outras, as seguintes ferramentas e tecnologias para essa transição em uma empresa:
I. É usado para verificar o código sem realmente executá-lo. Este tipo de ferramenta ajuda a encontrar vulnerabilidades em potencial no código-fonte, evitando que ocorram várias vulnerabilidades do tipo zero-day. Common Weakness Enumeration (CWE) é uma das classificações de avisos mais comuns produzidos por estas ferramentas. CWE é uma lista oficial ou dicionário de pontos fracos de segurança comuns exploráveis por invasores para obter acesso não autorizado ao sistema.
II. Da mesma forma que as ferramentas que executam testes de caixa preta, estes analisadores dinâmicos podem identificar vulnerabilidades do programa, como injeções de SQL, estouros de buffer e similares.

III. Este tipo de ferramenta analisa o comportamento do aplicativo, implementando uma análise de segurança contínua, sendo uma das tecnologias de segurança usadas em tempo de execução.

Os itens I, II e III correspondem, correta e respectivamente, a

a) RASP (Runtime Application Self-Protection) – SAST (Static Application Security Testing) – IAST (Interactive Application Security Testing).
b) IAST (Interactive Application Security Testing) – SAST (Static Application Security Testing) – DAST (Dynamic Application Security Testing).
c) SAST (Static Application Security Testing) – IAST (Interactive Application Security Testing) – DAST (Dynamic Application Security Testing).
d) IAST (Interactive Application Security Testing) – SAST (Static Application Security Testing) – RASP (Runtime Application Self-Protection).
e) SAST (Static Application Security Testing) – DAST (Dynamic Application Security Testing) – RASP (Runtime Application Self-Protection).

26. (FCC – TRT - 17ª Região (ES)– 2022) Para construir um pipeline como código no Jenkins, um analista utilizou um arquivo de texto simples conhecido como

a) Pipejenkins que especifica o encadeamento dos estágios do processo DevSecOps.
b) Scripted que especifica o enlace crítico entre as etapas primárias DevOps.
c) JenkinsFile que normalmente contém um código Groovy Domain Specific Language.
d) JenkinsGroovy que especifica o enlace crítico usando código DomainPline.
e) Declarative que especifica o enlace crítico usando código GroovyPline.

27. (IBFC – MGS– 2022) Quanto aos conceitos básicos de DevOps, segundo a Amazon, analise as afirmativas abaixo, dê valores Verdadeiro (V) ou Falso (F).

( ) Lança-se versões de software em ciclos periódicos, lineares, mas longos.
( ) As equipes de desenvolvimento ficam separadas dos operadores de software.
( ) DevOps é a combinação de filosofias culturais, práticas e ferramentas.

Assinale a alternativa que apresenta a sequência correta de cima para baixo.

a) V - F - F
b) V - V - F

c) F - V - V
d) F - F - V

28. (FUNDEP (Gestão de Concursos) – UFJF – 2022) Considere a sequência de comandos executados com sucesso em um repositório git para implementação de uma nova funcionalidade.
$ git branch cadastro-funcionario
$ git checkout cadastro-funcionario
...
$ git commit -a -m "Implementação do cadastro de funcionários"
$ git checkout master && git merge cadastro-funcionario
$ git push

Em relação à cultura DevOps e ao controle de versão, assinale a alternativa correta.

a) A funcionalidade desenvolvida foi registrada, mas não enviada ao servidor.
b) Foi utilizada a técnica de feature branch para implementação da funcionalidade.
c) O comando de merge foi aplicado no branch cadastro-funcionario.
d) O comando de commit pode não ter registrado todasasmudanças realizadasnaimplementação.
e) O comando de push foi realizado no branch cadastro-funcionario.

29. (FEPESE – FAPESC– 2022) Muitas organizações têm adotado práticas de DevOps no desenvolvimento de software.

Assinale a alternativa correta em relação ao assunto.
a) Baseado no modelo waterfall, DevOps é ideal para auxiliar equipes a manter um ritmo com modelos de desenvolvimento e entrega acelerados, tais como integração contínua e entrega para produção contínua (CI/CD).
b) Integração Contínua (Continuous integration - CI) é uma prática de desenvolvimento de software na qual desenvolvedores regularmente atualizam o seu código (realiza um commit) em um repositório compartilhado.
c) O uso de DevOps tem como objetivo criar uma cultura de colaboração entre as equipes de desenvolvimento e de operações que permite aumentar o fluxo de trabalho completado, reduzindo a quantidade de deploys, ao mesmo tempo aumentando a estabilidade e robustez do ambiente de produção.

d) A integração contínua (Continuous Integration - CI) foca na disponibilização de blocos de código completos para um repositório em intervalos regulares de tempo. Estes blocos de Código devem sempre estar em condições de serem executados para serem testados ou colocados em produção.
e) No uso de microservices, como uma arquitetura monolítica, CD permite aos desenvolvedores serem responsáveis por partes maiores e gerenciáveis do código que implementam funcionalidades individuais e trabalhar nestas em paralelo.

30. (IBADE – SEA-SC – 2022) Em um modelo DevOps existe um método para entregar aplicações com frequência aos clientes, visando integração, entrega e implantação contínuas. Chamamos esse método de:

a) git
b) pipeline
c) flowchart
d) reentrante
e) CI/CD

31. (UFAC – UFAC– 2022) O administrador DevOps acessou um servidor Linux da empresa, ao utilizar o comando "df" no "i" apontou 100% de uso. Julgue a informação recebida pelo console:

a) A memória swap atingiu sua capacidade máxima
b) A memória RAM não possui espaço disponível
c) O número de conexões máximas foi atingido
d) O sistema de arquivos raiz está cheio
e) Nenhuma das alternativas está correta.

GABARITO

1. Errado
2. Letra A
3. Letra A
4. Letra C
5. Letra C
6. Letra C
7. Letra B
8. Errado
9. Errado
10. Correto
11. Letra A
12. Correto
13. Errado
14. Correto
15. Letra D
16. Letra A
17. Letra D
18. Correto
19. Correto
20. Correto
21. Letra B
22. Letra A
23. Letra E
24. Letra D
25. Letra E
26. Letra C
27. Letra D
28. Letra B
29. Letra B
30. Letra E
31. Letra D

Sumário

Apresentação da Aula 2
Docker 3
Conceitos Básicos 3
Principais objetos Docker 6
Containers 8
Namespaces 9
Registry 11
Comparando Contêineres e Máquinas Virtuais 11
Arquivos Essenciais: Dockerfile e Arquivo de Composição 13
Dockerfile 13
FROM 17
RUN 18
CMD 21
LABEL 21
EXPOSE 21
ENV 22
ADD 22
COPY 22
VOLUME 23
WORKDIR 23
ARG 24
Arquivo de composição 25
Docker CLI 25
Comandos 25
Docker Compose 31
Principais tipos de rede Docker 34
Docker Swarm 39
Docker Desktop 40
Referências 41

APRESENTAÇÃO DA AULA

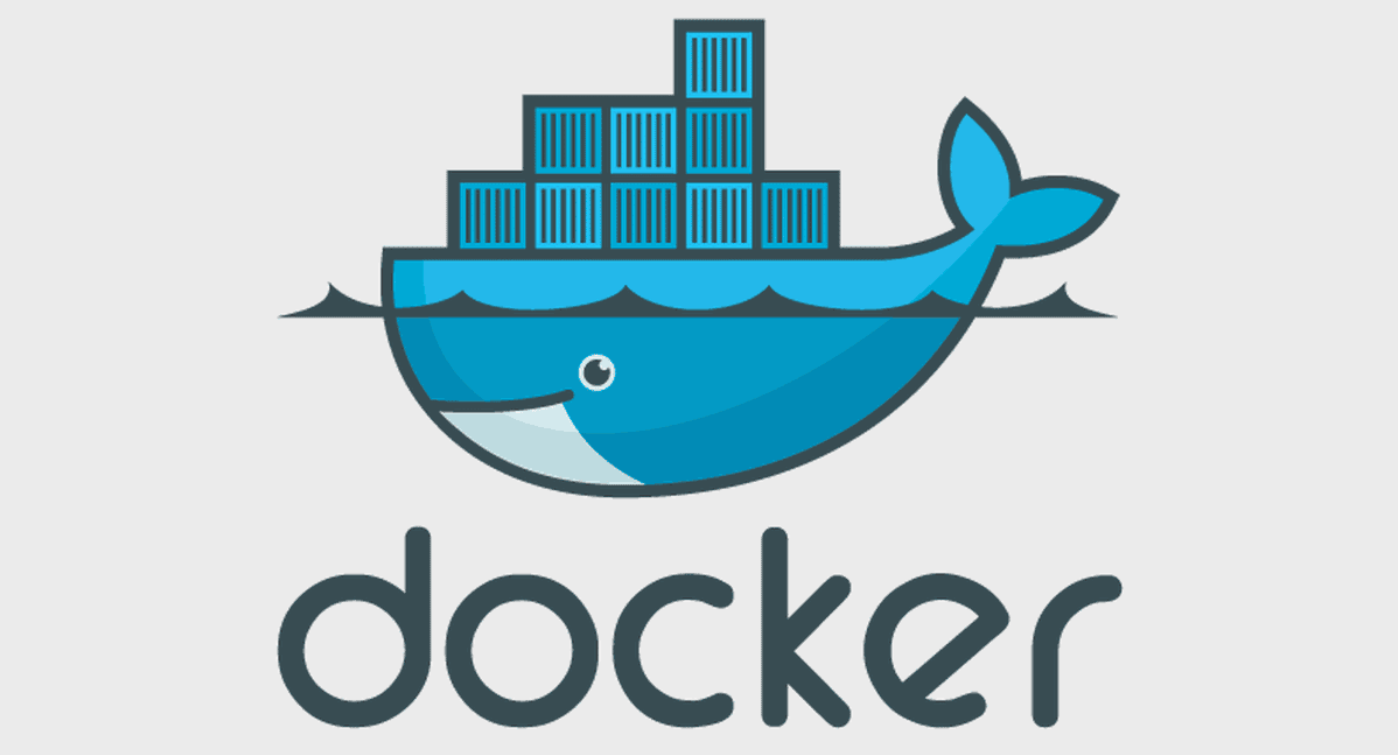
Olá, pessoal! Hoje vamos falar sobre DOCKER! 🐬 Nesta aula, vamos mergulhar no fascinante mundo dos contêineres e explorar conceitos básicos, principais objetos Docker e muito mais. Começaremos nossa jornada explorando os conceitos fundamentais do Docker, como o que é um contêiner, como ele se diferencia das máquinas virtuais e os benefícios que oferece. Em seguida, vamos aprofundar nossa compreensão dos principais objetos Docker, como containers, namespaces e o Docker Registry, onde imagens de contêineres são armazenadas e compartilhadas.

Não podemos esquecer de discutir os arquivos essenciais para criar e configurar contêineres no Docker. Vamos examinar o Dockerfile, que desempenha um papel crucial na criação de imagens personalizadas, e suas instruções, como FROM, RUN, CMD, LABEL, EXPOSE e muito mais. Também abordaremos o Arquivo de Composição, uma ferramenta poderosa para gerenciar contêineres complexos com o Docker Compose.

Ao longo da aula, exploraremos a interface de linha de comando do Docker e seus principais comandos. Além disso, discutiremos os principais tipos de rede Docker, o Docker Swarm para orquestração de contêineres e o Docker Desktop para facilitar o desenvolvimento local.

DOCKER

Conceitos Básicos



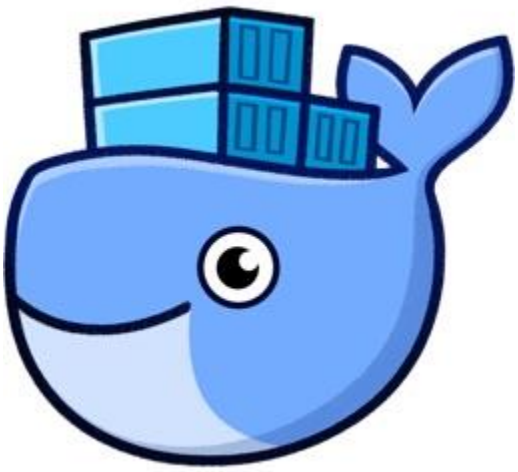
O Docker é uma **plataforma open source** para **desenvolver, enviar e executar aplicações**. O Docker permite que você separe suas aplicações da sua infraestrutura para que você possa entregar software rapidamente. Com o Docker, você pode **gerenciar sua infraestrutura** da mesma forma que você **gerencia suas aplicações**. Aproveitando as metodologias do Docker para enviar, testar e implantar código, você pode reduzir significativamente o atraso entre escrever código e executá-lo em produção.

O Docker oferece uma camada de abstração e automatização para a virtualização de sistemas operacionais no Windows e no Linux. O Docker consiste em um conjunto de produtos de plataforma como serviço que usam virtualização de nível de sistema operacional para entregar software em pacotes chamados contêineres. Os contêineres são isolados uns dos outros, agrupando seus próprios softwares, bibliotecas e arquivos de configuração.

Mas como exatamente o Docker realiza essa tarefa? De acordo com a documentação, o Docker utiliza técnicas como **isolamento de recursos do núcleo do Linux**, namespaces do núcleo e um sistema de arquivos que incorpora recursos de sobreposição (OverlayFS). Isso resulta na criação de contêineres independentes, os quais são executados dentro de uma única instância do sistema operacional. Essa abordagem evita a necessidade de lidar com as demandas pesadas de manutenção de máquinas virtuais (VMs).

O Docker se apresenta como uma **alternativa de virtualização** na qual o núcleo da máquina hospedeira é compartilhado com o software ou a máquina virtualizada em operação. Virtualização é um termo amplo

que pode se referir a diferentes tecnologias. No caso do Docker, é mais preciso dizer que ele é uma tecnologia de virtualização de nível de sistema operacional, também conhecida como **virtualização de contêineres**. Isso possibilita que desenvolvedores incluam as bibliotecas e outras dependências do programa diretamente no software, com um impacto menor no desempenho em comparação à virtualização completa de um servidor. Dessa maneira, o Docker transforma operações em infraestruturas como serviços da web em processos mais intercambiáveis, eficientes e flexíveis.



Na prática, o Docker é utilizado como uma ferramenta para **empacotar um aplicativo e todas as suas dependências em um recipiente virtual**. Esse recipiente pode ser executado em qualquer servidor que possua o Docker instalado, proporcionando flexibilidade e portabilidade ao local de execução do aplicativo. Isso pode incluir ambientes locais, nuvens públicas, nuvens privadas, entre outras opções.

Assim, podemos concluir que o Docker é uma plataforma que **utiliza contêineres como base arquitetônica para aumentar a portabilidade e a eficiência das aplicações**.

Além disso, o Docker é uma alternativa de virtualização que oferece, entre outros, os seguintes benefícios:

- **Portabilidade**: Os contêineres podem ser executados em qualquer servidor que possua o Docker instalado, independentemente do sistema operacional ou da infraestrutura subjacente.
- **Eficiência**: Os contêineres compartilham o núcleo do sistema operacional com outros contêineres, o que reduz o uso de recursos e melhora o desempenho.
- **Flexibilidade**: Os contêineres podem ser facilmente criados, modificados e gerenciados, o que torna mais fácil para os desenvolvedores e administradores de TI entregar e escalar aplicativos.

O Docker é uma plataforma **open source** que facilita a **criação e administração de ambientes isolados**. O Docker pode ser usado para **empacotar uma aplicação ou ambiente inteiro** dentro de um container, ou apenas um aplicativo ou serviço. Os **containers** são **ambientes isolados** que agrupam um aplicativo e todas as suas dependências em uma única instância do sistema operacional. Isso permite que os desenvolvedores distribuam e executem aplicativos com mais eficiência e flexibilidade.

Ainda, ele pode ser usado para **empacotar aplicações e ambientes para implantação em qualquer ambiente que tenha o Docker instalado**. Isso reduz drasticamente o tempo de deploy, pois não há a necessidade de ajustes de ambiente para o correto funcionamento do serviço. O ambiente é sempre o mesmo, basta configurar uma vez e replicar quantas vezes quiser.

O Docker também pode ser usado para **criar imagens de forma automática**, usando ferramentas como o Docker Hub. Isso permite que os desenvolvedores criem imagens rapidamente e facilmente, sem precisar escrever um Dockerfile manualmente.

O Docker possui como propósito principal **permitir que você gerencie sua infraestrutura da mesma forma que gerencia seus aplicativos**, ou seja, através do conceito de "infraestrutura como código" (IaC). Além disso, ele possibilita o uso de imagens de ambientes completos para promover consistência entre diferentes

camadas de implantação, como produção (prd), homologação (hml) e desenvolvimento (dev), **garantindo a integridade do software em todas as fases do ciclo de vida**.

Vantagens:

- Agilidade no provisionamento de ambientes.
- Padronização dos ambientes.
- Implementação do conceito de infraestrutura como código.
- Requer recursos computacionais mínimos para funcionar.

O Docker Engine é composto por um aplicativo **cliente-servidor**, compreendendo os seguintes **elementos principais**:

1. **Um servidor de execução contínua**, que opera em segundo plano como um processo de serviço (chamado de "dockerd").

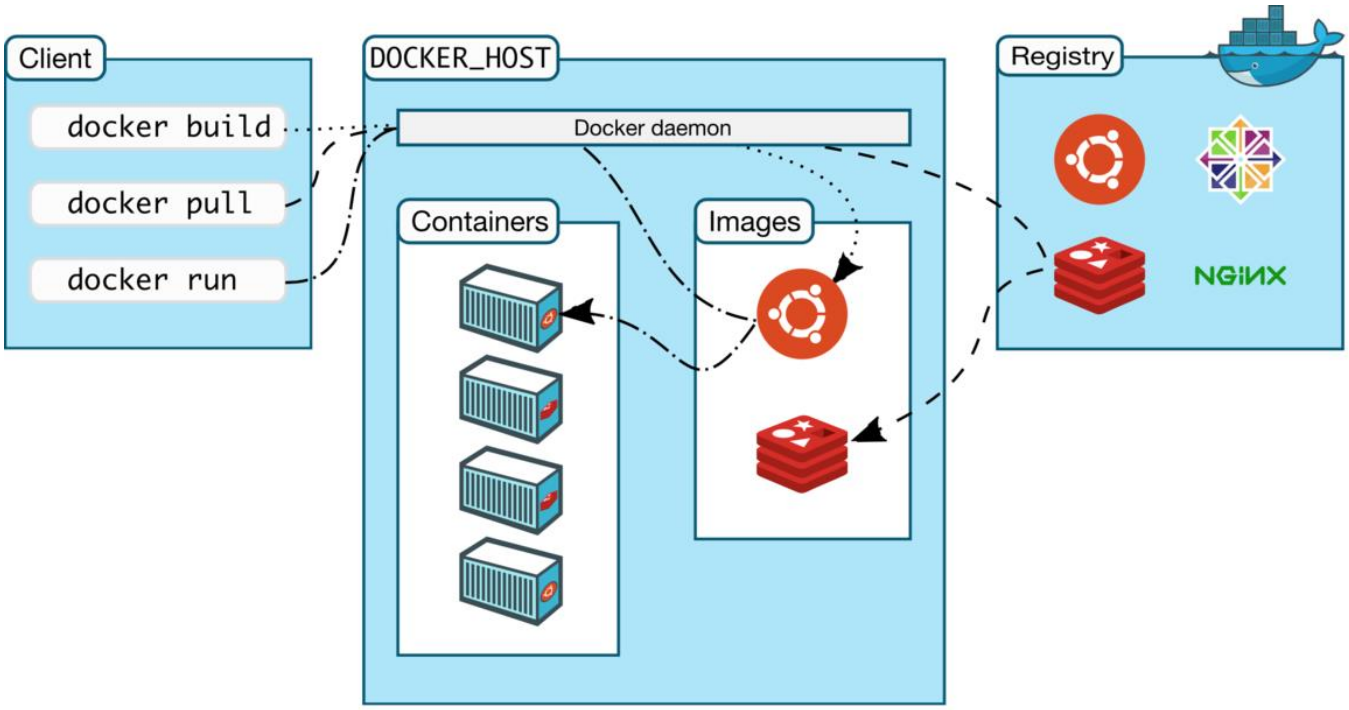
2. **Uma API REST** que define as interfaces pelas quais os programas podem se comunicar com o daemon e dar instruções.

3. **Um cliente de linha de comando (CLI)**, conhecido como "Comando docker".

A **CLI** utiliza a **API REST** do Docker para controlar ou interagir com o daemon do Docker, seja por meio de scripts ou comandos diretos na linha de comando. Muitos outros aplicativos relacionados ao Docker também utilizam essa API e a CLI como base.

O **daemon** é responsável pela criação e gerenciamento de entidades Docker, como imagens, contêineres, redes e volumes.

O Docker opera em uma **arquitetura cliente-servidor**, em que o cliente Docker interage com o **daemon do Docker** para realizar tarefas como a **criação, execução e distribuição de contêineres**. Essa interação pode ocorrer no mesmo sistema, ou é possível conectar um cliente Docker a um daemon Docker em uma máquina remota. A comunicação entre o cliente e o daemon do Docker é estabelecida por meio de uma **API REST**, que pode ser acessada através de **sockets UNIX** locais ou por uma **interface de rede**.



O daemon do Docker, conhecido como "dockerd", está constantemente atento às solicitações feitas através da API do Docker e é responsável por gerenciar os diversos objetos que compõem o ecossistema Docker, tais como imagens, contêineres, redes e volumes. Além disso, um daemon pode se comunicar com outros daemons para coordenar e administrar serviços Docker.

Por outro lado, o cliente do Docker, denominado "docker client", é a principal interface utilizada pela maioria dos usuários para interagir com a plataforma Docker. Quando você emite comandos como "docker run", o cliente Docker os encaminha para o daemon Docker (dockerd) que, por sua vez, executa essas instruções. O comando "docker" aproveita a funcionalidade fornecida pela API do Docker, e o cliente do Docker tem a capacidade de se comunicar com múltiplos daemons, se necessário.

Os Registros Docker, como o Docker Hub, funcionam como repositórios para armazenar imagens Docker. O Docker Hub é um registro público acessível a qualquer pessoa por padrão, e o Docker é configurado para procurar imagens neste registro por padrão. Contudo, é possível configurar e operar seu próprio registro privado, e no caso do Docker Datacenter (DDC), ele inclui o DTR (Docker Trusted Registry), que oferece recursos adicionais para gerenciamento e segurança de imagens Docker.

Principais objetos Docker

Imagine uma **imagem** como um **modelo de uma máquina virtual**, mas de maneira mais eficiente. É uma espécie de "receita" que **diz ao Docker como criar um container**. Essas imagens são como blocos de construção. Você pode pegar uma imagem básica, como uma página em branco, e adicioná-la com todas as coisas que você precisa. Por exemplo, você pode começar com uma imagem básica que contenha apenas o sistema operacional, como se fosse uma folha em branco, e depois adicionar a ela tudo o que seu aplicativo precisa para funcionar corretamente, como um servidor web e um banco de dados.

Agora, pense em um **contêiner** como uma **instância em execução dessa imagem**. É como se você tivesse construído uma máquina virtual usando a receita da imagem. Essa máquina virtual é um **contêiner**, e você

pode ter **várias cópias** dele em execução ao mesmo tempo, todas baseadas na mesma imagem, mas cada uma delas pode ser personalizada de forma diferente. Esses contêineres são isolados uns dos outros e do sistema host, o que significa que podem rodar suas próprias versões de aplicativos sem afetar uns aos outros ou o sistema onde estão rodando.

Suponha que você está criando um site da web. Você pode começar com uma imagem básica que contém um sistema operacional, como se fosse uma página em branco, e, em seguida, você adiciona a essa imagem tudo o que é necessário para executar seu site, como um servidor web, código-fonte, banco de dados e assim por diante. Agora, você pode criar vários contêineres baseados nessa imagem, cada um deles representando uma instância do seu site. Eles compartilham a mesma base (a imagem), mas podem ser configurados de maneira diferente, por exemplo, para ter versões diferentes do seu site em execução.

Portanto, **imagens são como modelos e contêineres são as instâncias em execução** desses modelos. Eles permitem que você crie e execute aplicativos de maneira isolada e eficiente, tornando o desenvolvimento e implantação mais rápidos e flexíveis.

(FGV – SEFAZ-AM – 2022) Leia o fragmento a seguir.
"A plataforma Docker usa uma arquitetura do tipo _____. O cliente Docker conversa com o daemon do Docker, que constrói, executa e distribui _____ Docker. O cliente e o daemon do Docker podem ser executados em um mesmo sistema ou se conectar um cliente do Docker a um daemon remoto. O cliente Docker e o daemon se comunicam usando _____ ou uma interface de redes."

Assinale a opção cujos itens completam corretamente as lacunas do fragmento acima, na ordem apresentada.

a) MVC - imagens - chamadas RPC ou bluetooth.
b) thin client - contêineres - wireless ou bluetooth.
c) serverless - componentes - chamadas MPI ou RPC.
d) cliente-servidor - contêineres - API REST ou soquetes UNIX.
e) mesh app and service - imagens - API RESTFULL ou wireless.

Comentários:

Pessoal, o fragmento menciona o funcionamento da plataforma Docker e descreve como os diferentes componentes interagem. Vamos analisar cada parte do fragmento:

"A plataforma Docker usa uma arquitetura do tipo _____."
Aqui, o texto se refere à arquitetura subjacente que o Docker utiliza. O Docker segue uma arquitetura cliente-servidor. "O cliente Docker conversa com o daemon do Docker, que constrói, executa e distribui _____ Docker." O cliente Docker é a interface por meio da qual os usuários interagem com o Docker. Ele se comunica com o daemon Docker, que é o processo responsável por gerenciar os containers. O daemon constrói, executa e distribui imagens Docker.

"O cliente e o daemon do Docker podem ser executados em um mesmo sistema ou se conectar a um cliente do Docker a um daemon remoto." Isso significa que tanto o cliente quanto o daemon podem estar no mesmo sistema (máquina) ou podem estar em sistemas diferentes, permitindo a conexão a um daemon remoto, se necessário.

"O cliente Docker e o daemon se comunicam usando _____ ou uma interface de redes." Aqui, o texto aborda os métodos de comunicação entre o cliente Docker e o daemon Docker. Eles se comunicam usando uma API REST (Representational State Transfer) ou soquetes UNIX.

A opção correta que preenche corretamente as lacunas do fragmento é:

d) cliente-servidor - contêineres - API REST ou soquetes UNIX.

Portanto, o Docker opera em uma arquitetura cliente-servidor, onde o cliente Docker se comunica com o daemon Docker. O daemon constrói, executa e distribui contêineres Docker. A comunicação entre o cliente e o daemon é realizada por meio de uma API REST ou soquetes UNIX, dependendo do cenário. (**Gabarito**: Letra D)

Containers

Um contêiner é uma **unidade padronizada de software** que **empacota código** e todas as suas dependências, permitindo que a **aplicação seja executada de forma rápida e confiável** em diversos ambientes computacionais. Uma imagem de contêiner Docker é um **pacote leve, independente e executável** de software que inclui tudo o que é necessário para executar uma aplicação: código, tempo de execução, ferramentas do sistema, bibliotecas do sistema e configurações.

Empacotar software em unidades padronizadas para desenvolvimento, envio e implantação é uma prática essencial na moderna gestão de aplicações. Utilizar contêineres é uma abordagem eficiente para alcançar essa padronização. Os contêineres oferecem uma série de **vantagens**, tornando-os ideais para o ciclo de vida de uma aplicação. Eles são **flexíveis, permitindo a contêinerização** de até mesmo os aplicativos mais complexos, tornando sua gestão mais eficaz. Além disso, são **leves**, uma vez que compartilham o kernel do sistema operacional hospedeiro, economizando recursos. Sendo **intercambiáveis e portáteis**, os contêineres podem ser atualizados facilmente e executados em qualquer ambiente, seja localmente ou na nuvem. A escalabilidade automática e a capacidade de empilhamento vertical de serviços tornam os contêineres uma escolha poderosa para desenvolvedores e operadores de TI.

Os contêineres são a base para a **criação, compartilhamento e execução eficiente de aplicações**. Sua **flexibilidade, leveza e capacidade de serem intercambiáveis** fazem com que sejam uma escolha ideal para abordar os desafios modernos de desenvolvimento e implantação de software. Eles permitem que as aplicações sejam desenvolvidas de maneira **padronizada**, empacotadas de forma independente e executadas consistentemente em diversos ambientes. Além disso, a **escalabilidade automática** e a **portabilidade** dos contêineres facilitam a adaptação das aplicações às necessidades em constante mudança.

Os contêineres são criados a partir de imagens de contêineres durante a execução, e no caso dos contêineres Docker, essas imagens se transformam em contêineres quando são executadas no Docker Engine. Essa abordagem é aplicável tanto a aplicativos baseados em Linux quanto em Windows, garantindo que o software em contêiner sempre funcione da mesma forma, independentemente da infraestrutura subjacente. Os contêineres oferecem isolamento para o software, assegurando que ele opere de maneira consistente, mesmo em cenários onde existam diferenças, como entre os ambientes de desenvolvimento e preparo.

Os contêineres Docker que são executados no **Docker Engine** seguem um conjunto de princípios fundamentais que os tornam uma escolha poderosa. São considerados **padrão** na indústria de contêineres, garantindo portabilidade em qualquer ambiente. Sendo leves, os contêineres **compartilham o kernel do sistema operacional da máquina**, eliminando a necessidade de um sistema operacional dedicado para cada aplicação. Isso resulta em uma utilização mais eficiente dos servidores e redução de custos com servidores e licenciamento. Além disso, os contêineres oferecem **segurança sólida**, com o Docker proporcionando **as capacidades de isolamento mais robustas da indústria**, tornando as **aplicações mais seguras** quando executadas em contêineres.

Os contêineres Docker têm uma presença ubíqua, estando disponíveis em diversas plataformas, incluindo Linux, Windows, data centers, nuvem e até mesmo ambientes serverless. Essa onipresença é possível porque o Docker aproveitou conceitos de computação existentes relacionados a contêineres, especialmente no mundo Linux, onde são usadas primitivas conhecidas como **cgroups e namespaces**. A tecnologia do Docker é única devido ao seu foco nas necessidades de desenvolvedores e operadores de sistemas, permitindo a separação das dependências de aplicativos da infraestrutura subjacente.

O sucesso no mundo Linux levou a uma parceria com a Microsoft, que trouxe os contêineres Docker e sua funcionalidade para o Windows Server. Além disso, a tecnologia disponibilizada pelo Docker e seu projeto de código aberto, o Moby, foi adotada por todos os principais fornecedores de data centers e provedores de nuvem. Muitos desses provedores utilizam o Docker em suas ofertas de infraestrutura como serviço (IaaS) nativas de contêineres. Além disso, os principais frameworks open source para ambientes serverless também fazem uso da tecnologia de contêineres Docker, demonstrando sua versatilidade e ampla aceitação no cenário atual de desenvolvimento de software.

Namespaces

O Docker utiliza uma tecnologia chamada namespaces para fornecer o espaço de trabalho isolado chamado de contêiner. Quando você executa um contêiner, o Docker cria um conjunto de namespaces para aquele contêiner. Esses namespaces oferecem uma camada de isolamento. Cada aspecto de um contêiner é executado em um namespace separado e seu acesso é limitado a esse namespace.

PID namespace

Imagine que você tem um computador com vários programas em execução. Cada programa tem um identificador exclusivo, chamado PID, que o identifica no sistema operacional. O PID namespace é um recurso que permite dividir esses identificadores de processos em diferentes "espaços". Cada espaço de PID tem seus próprios PIDs exclusivos, e os PIDs de um espaço de PID não podem ser vistos ou acessados por programas em outros espaços de PID.

No contexto do Docker, o PID namespace é usado para isolar os processos de um container dos processos do host. Isso significa que os processos de um container não podem ver ou acessar os processos do host, e vice-versa. Essa separação de processos é importante para a segurança e a confiabilidade dos containers. Ela ajuda a impedir que os processos de um container sejam afetados por problemas nos processos do host.

Por exemplo, se um container está executando um programa malicioso, esse programa não poderá acessar ou controlar outros processos no sistema.

O espaço de identificação de processos é um conjunto de identificadores exclusivos que são usados para identificar processos no sistema operacional. O PID namespace permite que cada container tenha seu próprio espaço de identificação de processos, o que significa que os processos de um container não podem ser vistos ou acessados por processos de outros containers. O PID namespace é uma ferramenta importante para a segurança e a confiabilidade dos containers. Ele ajuda a impedir que os processos de um container sejam afetados por problemas em outros containers.

Net Namespace

O Net Namespace permite que cada container possua sua interface de rede e portas. **Para que seja possível a comunicação entre os containers, é necessário criar dois Net Namespaces diferentes**, um responsável pela interface do container (normalmente utilizamos o mesmo nome das interfaces convencionais do Linux, por exemplo, a etho) e outro responsável por uma interface do host, normalmente chamada de veth* (veth + um identificador aleatório). Essas duas interfaces estão linkadas através da bridge Dockero no host, que permite a comunicação entre os containers através de roteamento de pacotes.

Quando um Net Namespace é criado, ele é atribuído a um container. O container então usa a interface de rede, o endereço IP e o conjunto de portas do Net Namespace para se comunicar com outros containers, hosts e a internet.

A comunicação entre containers ocorre através de um processo chamado de encapsulamento. Quando um container envia um pacote de dados para outro container, o pacote é encapsulado com o endereço IP e o conjunto de portas do Net Namespace do container de destino. O pacote é então enviado para a interface de rede do container de origem.

A interface de rede do container de origem envia o pacote para a ponte Dockero. A ponte Dockero é uma ponte virtual que é criada automaticamente pelo Docker quando um container é criado. A ponte Dockero recebe pacotes de todas as interfaces de rede dos containers e encaminha os pacotes para o destino apropriado.

No caso de dois containers que estão no mesmo Net Namespace, a ponte Dockero não é necessária. Os pacotes de dados são simplesmente enviados diretamente da interface de rede de um container para a interface de rede do outro container. No caso de dois containers que estão em Net Namespaces diferentes, a ponte Dockero é necessária para encaminhar os pacotes de dados entre os containers. A ponte Dockero usa o endereço IP e o conjunto de portas dos Net Namespaces para determinar o destino do pacote de dados.

Os Net Namespaces podem ser usados para controlar a comunicação entre containers de várias maneiras. Por exemplo, os Net Namespaces podem ser usados para:

- Permitir que dois containers se comuniquem entre si, mas não com outros containers.
- Permitir que dois containers se comuniquem entre si, mas apenas em um determinado intervalo de portas.
- Permitir que dois containers se comuniquem entre si, mas apenas a partir de um determinado endereço IP.

Os Net Namespaces também podem ser usados para melhorar a segurança. Por exemplo, se um container é comprometido, ele não poderá se comunicar com outros containers que não estão no mesmo Net Namespace.

Registry

Um registro (**Registry**) é um aplicativo de servidor sem estado (stateless) e altamente escalonável que armazena e permite distribuir imagens Docker. O Registro é de código aberto, sob a licença permissiva Apacheopen_in_new.

O Registry oferece um meio **centralizado** de armazenar e distribuir imagens de contêiner Docker, garantindo que você tenha controle total sobre a localização do armazenamento, a distribuição eficiente e a integração perfeita com seu ambiente de desenvolvimento.

Essas imagens são utilizadas como base para a criação de contêineres Docker. Um registro é essencialmente um repositório de imagens que pode ser público, privado ou hospedado localmente.

Existem registros públicos, como o Docker Hub, que permitem que qualquer pessoa acesse e baixe imagens de contêiner compartilhadas pela comunidade. Além disso, muitas organizações configuram registros privados para armazenar suas imagens internamente, permitindo um controle mais granular sobre quem pode acessar e modificar essas imagens.

O Docker CLI (Interface de Linha de Comando) permite que os desenvolvedores enviem, busquem e gerenciem imagens de contêiner em registros. Isso torna o processo de compartilhamento e implantação de imagens Docker mais eficiente e escalável em ambientes de desenvolvimento, teste e produção.

Comparando Contêineres e Máquinas Virtuais

Contêineres e máquinas virtuais oferecem benefícios semelhantes de isolamento e alocação de recursos, mas operam de maneira diferente, pois os contêineres virtualizam o sistema operacional em vez do hardware subjacente. Os contêineres são mais portáteis e eficientes.

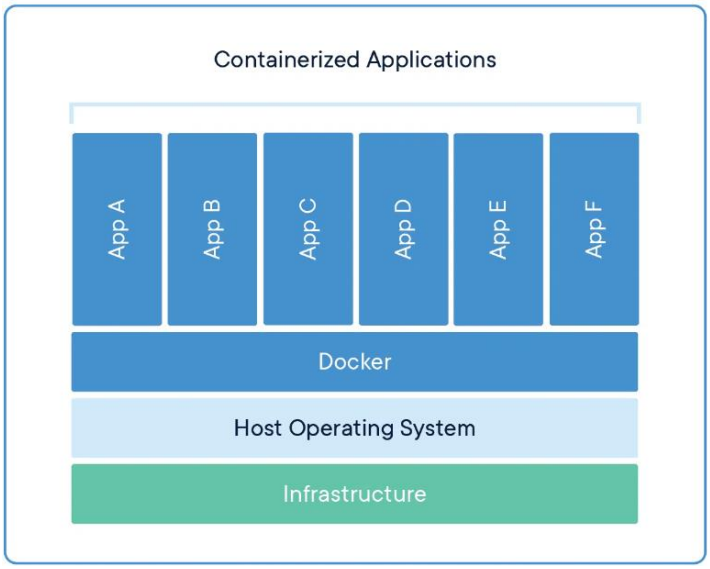
Contêineres

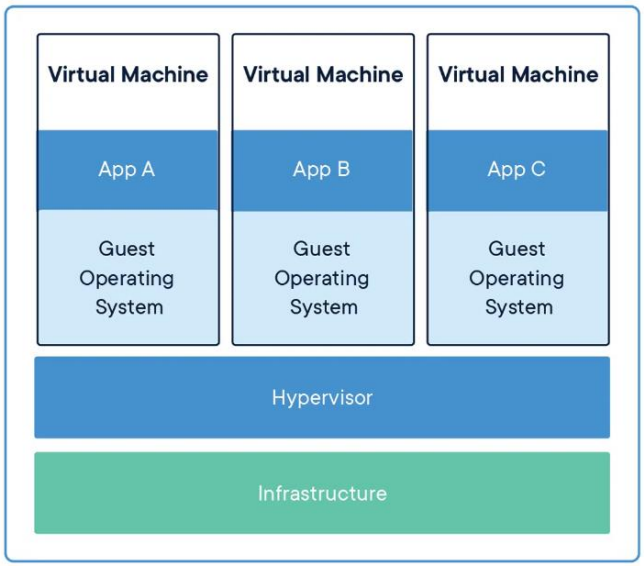
Contêineres são uma abstração no nível de aplicação que empacota código e suas dependências juntas. Múltiplos contêineres podem ser executados na mesma máquina e compartilhar o kernel do sistema operacional com outros contêineres, cada um executando como processos isolados no espaço do usuário. Os contêineres ocupam menos espaço do que as máquinas virtuais (as imagens de contêiner geralmente

têm dezenas de MBs de tamanho), podem lidar com mais aplicativos e requerem menos máquinas virtuais e sistemas operacionais.

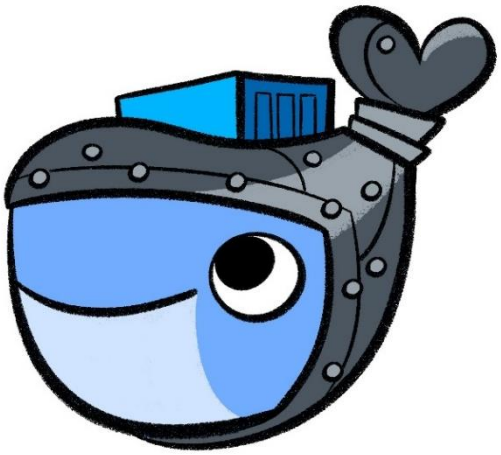
Máquinas Virtuais

Máquinas virtuais (VMs) são uma abstração do hardware físico, transformando um servidor em muitos servidores virtuais. O hipervisor permite que várias VMs sejam executadas em uma única máquina. Cada VM inclui uma cópia completa de um sistema operacional, a aplicação, binários e bibliotecas necessárias, ocupando dezenas de GBs de espaço. As VMs também podem ser lentas para inicializar.





Arquivos Essenciais: Dockerfile e Arquivo de Composição



Uma parte fundamental da virtualização de contêineres reside na capacidade de criar e gerenciar contêineres de forma consistente e repetível. Para alcançar esse feito, você precisa se familiarizar com dois tipos cruciais de arquivos: o Dockerfile e o arquivo de composição.

O Docker tem a capacidade de gerar imagens automaticamente ao interpretar as diretrizes presentes em um Dockerfile. Dockerfile, sendo um arquivo de texto, engloba **todas as instruções** que um utilizador poderia inserir na linha de comando para construir uma imagem.

Dockerfile contém todos os comandos que um usuário pode chamar na linha de comando para montar uma imagem.

Dockerfile

Um Dockerfile é um documento de texto que contém as instruções para configurar um ambiente para um contêiner Docker. É um arquivo de texto simples sem extensão de arquivo. O Dockerfile é usado para criar uma imagem Docker, que é um pacote de software leve, independente e executável que inclui tudo o que é necessário para executar um aplicativo: código, tempo de execução, ferramentas do sistema, bibliotecas do sistema e configurações.

No Dockerfile, você define desde a imagem base que será usada até a instalação de pacotes, configurações de rede, variáveis de ambiente e muito mais. A combinação dessas instruções resulta em um contêiner configurado e pronto para executar sua aplicação. O Dockerfile é a espinha dorsal do processo de construção e é o lugar onde você personaliza seu ambiente de contêiner.

Tudo bem, pessoal, já sabemos que um Dockerfile é como uma receita para criar um contêiner Docker. Ou seja, um arquivo de texto que contém instruções sobre como construir o seu próprio contêiner personalizado. Vamos agora dar uma olhada nas partes principais do formato:

O Dockerfile possui comentários, instruções e argumentos. É possível adicionar comentários para explicar o que está acontecendo no Dockerfile. Eles começam com um símbolo de hashtag (#) e **não afetam o funcionamento do Dockerfile**. São úteis para que você e outras pessoas entendam o que cada parte do Dockerfile faz.

Parser Directives

As Parser Directives são instruções opcionais que afetam a forma como as linhas subsequentes em um Dockerfile são tratadas. Elas não adicionam camadas à construção (build) e não aparecerão como uma etapa de construção no processo. As Parser Directives são escritas como um tipo especial de comentário no formato # directive=value. Cada diretiva pode ser usada apenas uma vez.

Uma vez que um comentário, linha em branco ou instrução do construtor tenha sido processada, o Docker não procura mais por Parser Directives. Em vez disso, qualquer coisa formatada como uma Parser Directive é tratada como um comentário, e o Docker não tenta validar se poderia ser uma Parser Directive. Portanto, todas as Parser Directives devem estar no início do Dockerfile.

Aqui estão algumas características importantes das Parser Directives:

Não diferenciam maiúsculas de minúsculas: As Parser Directives não são sensíveis a maiúsculas/minúsculas, mas é uma convenção escrevê-las em minúsculas.

Inclua uma linha em branco após as Parser Directives: É uma convenção incluir uma linha em branco após qualquer Parser Directive.

Não suportam caracteres de continuação de linha: Você não pode usar caracteres de continuação de linha nas Parser Directives.

Devido a essas regras, os seguintes exemplos são todos inválidos:

Inválido devido a caracteres de continuação de linha:

```
# direc \
tive=value
```

Inválido devido a aparecer duas vezes:

```
# directive=value1
# directive=value2
```

Tratado como um comentário devido a aparecer após uma instrução de construtor:

```
FROM ImageName
# directive=value
```

Tratado como um comentário devido a aparecer após um comentário que não é uma Parser Directive:

```
# Sobre o meu Dockerfile
# directive=value
FROM ImageName
```

As Parser Directives são usadas principalmente para configurar opções específicas do parser do Docker, como a sintaxe a ser usada ou a forma como os caracteres de escape são tratados. Atualmente, as Parser Directives suportadas são syntax e escape.

Variáveis de ambiente

As variáveis de ambiente são definidas no Dockerfile com o comando ENV. Elas podem ser usadas em várias instruções do Dockerfile, como RUN, COPY e ADD. Elas são definidas no Dockerfile com os símbolos $ ou ${. Elas são tratadas de forma equivalente e a sintaxe de chaves é geralmente usada para lidar com problemas com nomes de variáveis que não têm espaços em branco, como ${foo}_bar.

A sintaxe ${variable_name} também suporta alguns dos modificadores bash padrão conforme especificado abaixo:

- ${variable:-word} indica que, se a variável estiver definida, o resultado será esse valor. Se a variável não estiver definida, a palavra será o resultado.
- ${variable:+word} indica que, se a variável estiver definida, a palavra será o resultado, caso contrário, o resultado será a string vazia. Em todos os casos, a palavra pode ser qualquer string, incluindo variáveis de ambiente adicionais. É possível escapar da variável adicionando um \ antes dela: \$foo ou \${foo}, por exemplo, serão traduzidos para os literais $foo e ${foo}, respectivamente.

Variáveis de ambiente são suportadas pelas seguintes instruções na Dockerfile:

- ADD
- COPY
- ENV
- EXPOSE
- FROM
- LABEL
- STOPSIGNAL
- USER
- VOLUME
- WORKDIR
- ONBUILD (quando combinado com uma das instruções suportadas acima)

Arquivo .dockerignore

Antes de o CLI do Docker enviar o contexto ao daemon do Docker, ele procura um arquivo chamado .dockerignore no diretório raiz do contexto. Se esse arquivo existir, o CLI modifica o contexto para excluir arquivos e diretórios que correspondam aos padrões nele. Isso ajuda a evitar o envio desnecessário de arquivos e diretórios grandes ou confidenciais ao daemon e potencialmente adicioná-los às imagens usando ADD ou COPY.

O CLI interpreta o arquivo .dockerignore como uma lista de padrões separados por nova linha, semelhantes aos globs de arquivos das shells Unix. Para fins de correspondência, a raiz do contexto é considerada como o diretório de trabalho e a raiz. Por exemplo, os padrões /foo/bar e foo/bar excluem um arquivo ou diretório chamado bar no subdiretório foo de PATH ou na raiz do repositório git localizado em URL. Nenhum deles exclui mais nada.

Se uma linha no arquivo .dockerignore começa com # na coluna 1, essa linha é considerada um comentário e é ignorada antes de ser interpretada pelo CLI.

As **instruções** são como comandos que dizem ao Docker **o que fazer**. Elas são escritas em letras **maiúsculas** (embora não sejam sensíveis a maiúsculas/minúsculas). **As instruções são executadas em ordem, de cima para baixo, à medida que o Docker constrói o contêiner**.

Por fim, as instruções podem ter argumentos que fornecem informações adicionais. Por exemplo, quando você usa a instrução **FROM**, você precisa especificar de qual imagem você está construindo o seu contêiner. Essa informação é um argumento. É possível pensar nos argumentos como os ingredientes específicos da sua receita.

Falamos sobre a instrução FROM, mas o que ela faz? A instrução FROM especifica a imagem base para o build. Há inúmeras instruções usadas no Dockerfile. Vejamos algumas instruções.

Instrução	Descrição
FROM	Especifica a imagem base para o build (a construção da imagem). A imagem base fornece o código, as bibliotecas e as ferramentas necessárias para executar o aplicativo na imagem.
RUN	Executa um comando na imagem de construção.
COPY	Copia um arquivo ou diretório para a imagem de construção.
ADD	Adiciona um arquivo ou diretório à imagem de construção, descompactando-o se necessário.
ENV	Define uma variável de ambiente na imagem de construção. As variáveis de ambiente podem ser usadas nos comandos do Dockerfile ou no container em execução.
LABEL	Adiciona uma etiqueta à imagem de construção.
EXPOSE	A instrução EXPOSE informa ao Docker que o container escuta nas portas de rede especificadas em tempo de execução. Isso permite que os outros containers ou hosts se conectem ao container.
WORKDIR	Define o diretório de trabalho para os comandos subsequentes. O diretório de trabalho é o diretório a partir do qual os comandos são executados.
CMD	Define o comando que será executado quando a imagem for criada.
ENTRYPOINT	Define o comando que será executado quando a imagem for iniciada.

VOLUME	Define um diretório na imagem que pode ser compartilhado com o host. Isso pode ser útil para armazenar dados que precisam ser acessíveis a partir do host, como logs ou arquivos de configuração.

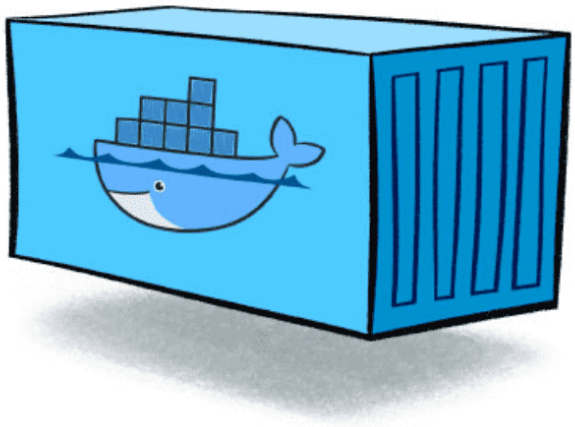
(CCV-UFC – UFC – 2019) Dockerfile é um um arquivo de texto que contém todos os comandos, em ordem, necessários para construir uma determinada imagem Docker. Sobre as instruções contidas em um Dockerfile, assinale a alternativa correta.

a) A instrução VOLUME configura o tamanho da imagem.
b) A instrução ENV adiciona metadados para uma imagem.
c) A instrução WORKDIR permite a criação de um diretório no host onde ficam armazenados os dados do container.
d) A instrução EXPOSE informa ao Docker que o container escuta nas portas de rede especificadas em tempo de execução.
e) A instrução FROM configura qual será a aplicação principal do container, sendo executada após a inicialização do container.

Comentários:

A alternativa correta é (d). A instrução EXPOSE informa ao Docker que o container escuta nas portas de rede especificadas em tempo de execução. As outras alternativas estão incorretas vejamos: A instrução VOLUME não configura o tamanho da imagem, mas sim cria um diretório na imagem que pode ser compartilhado com o host. A instrução ENV adiciona variáveis de ambiente para a imagem. A instrução WORKDIR define o diretório de trabalho para os comandos subsequentes na imagem. A instrução FROM especifica a imagem base para a construção da imagem. (**Gabarito**: Letra D)

FROM



A instrução "FROM" no Docker é como o ponto de partida quando você está construindo um contêiner. É como escolher um modelo para construir uma casa. Você precisa dizer de onde você quer começar. Neste caso, você deve começar com uma instrução "FROM". Esta instrução diz ao Docker qual imagem usar como base para o seu contêiner. Essa imagem pode ser qualquer uma que seja válida, e geralmente é fácil começar usando imagens disponíveis publicamente.

Portanto, para criar um Dockerfile válido, você deve começar com a instrução "FROM" para especificar a imagem base que será usada como ponto de partida para a construção do seu contêiner. Isso é fundamental para a construção bem-sucedida do seu contêiner, assim como escolher o tipo de solo certo é fundamental para construir uma casa forte.

O comando ARG é o único que pode vir antes do comando FROM no Dockerfile. O comando FROM pode ser usado várias vezes em um único Dockerfile para criar várias imagens ou usar um estágio de construção como dependência para outro. Para isso, basta anotar o último ID da imagem gerado pelo commit antes de cada novo comando FROM. Cada comando FROM limpa qualquer estado criado por comandos anteriores.

Opcionalmente, é possível dar um nome a um novo estágio de construção adicionando AS nome ao comando FROM. Esse nome pode ser usado em comandos subsequentes de FROM e COPY --from=<nome> para se referir à imagem construída neste estágio.

Os valores de tag ou digest são opcionais. Se você omitir um deles, o construtor assume automaticamente a tag "latest". O construtor retornará um erro se não conseguir encontrar o valor da tag.

É possível usar a opção --platform para especificar a plataforma da imagem quando o comando FROM faz referência a uma imagem multiplataforma. Essas plataformas podem ser: linux/amd64, linux/arm64 ou windows/amd64. Por padrão, a plataforma de destino da solicitação de construção é usada.

Além disso, é possível utilizar argumentos de construção globais no valor desta opção. Por exemplo, argumentos de plataforma automáticos permitem que você force um estágio a ser construído na plataforma nativa (--platform=$BUILDPLATFORM) e use essa plataforma para compilar cruzadamente para a plataforma de destino dentro desse estágio.

RUN

O comando RUN tem duas formas:

- Forma shell: é possível usar o comando RUN seguido de um comando entre aspas, como RUN <comando> (por padrão, ele é executado em um shell, que é /bin/sh -c no Linux ou cmd /S /C no Windows).

- Forma exec: Alternativamente, é possível usar o comando RUN na forma executável, assim: RUN ["executável", "param1", "param2"].

Quando você usa o comando RUN com um comando, ele executa esse comando em uma nova camada, como criar uma nova página em um caderno de receitas. Em seguida, ele salva os resultados desse comando na camada. Imagine isso como escrever o resultado de cada passo da receita na página correspondente.

A imagem resultante, que agora inclui todas as camadas com os comandos executados, é usada para a próxima etapa no Dockerfile, assim como você segue os passos em uma receita de bolo. Isso é ótimo porque é possível criar sua imagem passo a passo, e as camadas são como marcos ao longo do caminho. Se você quiser, pode voltar a qualquer ponto da história da imagem, assim como folhear seu caderno de receitas para encontrar uma receita anterior.

(FGV – TRT - 16ª REGIÃO (MA) – 2022) Docker é uma plataforma que permite criar e compartilhar aplicativos e microsserviços em contêineres.

O comando utilizado para executar um contêiner novo é o

a) docker composes.
b) docker create.
c) docker build.
d) docker exec.
e) docker run.

Comentários:

A resposta correta é a letra E. O comando docker run é usado para executar um contêiner novo. Ele pode ser usado para iniciar um contêiner a partir de uma imagem existente ou para construir e iniciar um contêiner a partir de um Dockerfile.

As outras opções não são adequadas para este cenário. O comando docker compose é usado para iniciar um conjunto de contêineres. O comando docker create é usado para criar um contêiner, mas não o inicia. O comando docker build é usado para construir uma imagem Docker. O comando docker exec é usado para executar um comando em um contêiner já em execução.

Gabarito: Letra E

Além disso, há duas maneiras de usar o comando RUN: a forma shell, onde você executa comandos em um shell padrão, e a forma exec, que permite executar comandos diretamente, o que é útil quando você não quer usar um shell específico. Por fim, é possível até mesmo personalizar o shell padrão se quiser, usando o comando SHELL. Docker -run suporta várias opções que permitem personalizar a execução do contêiner.

Opção	Abreviação	Descrição
--attach	-a	Anexar à entrada padrão (STDIN), saída padrão (STDOUT) ou fluxo de erros padrão (STDERR).
--cpu-shares	-c	Ações de CPU (peso relativo), entre 10 e 1000.
--detach	-d	**Executar o contêiner em segundo plano** e imprimir o ID do contêiner.
--env	-e	Definir variáveis de ambiente.
--hostname	-h	Nome do host do contêiner.
--interactive	-i	Manter a entrada padrão (STDIN) aberta, mesmo se não estiver anexado.
--label	-l	Definir metadados em um contêiner.
--memory	-m	Limite de memória.
--name		Atribuir um nome ao contêiner
--publish	-p	Publicar a(s) porta(s) do contêiner no host.
--quiet	-q	Suprimir a saída da operação "pull" (baixar imagem).
--tty	-t	Alocar um pseudoterminal (TTY).

--user	-u	Nome de usuário ou UID (formato: <nome
--volume	-v	Montar um volume.
--workdir	-w	Diretório de trabalho dentro do contêiner.

(UFPR – IF-PR – 2023) O comando docker run cria e executa um novo contêiner a partir de uma imagem. Para a criação desse contêiner em segundo plano, utiliza-se o comando:

a) docker container run -b
b) docker container run -p
c) docker container run -a
d) docker container run -d
e) docker container run -s

Comentários:

O comando "docker run" é usado para criar e executar um novo contêiner a partir de uma imagem. Para executar o contêiner em segundo plano, ou seja, em modo daemon (background), utiliza-se a opção "-d". Portanto, a resposta correta é a opção "d) docker container run -d". Isso permite que o contêiner seja executado em segundo plano, liberando o terminal para que você possa continuar a usar a linha de comando para outras tarefas. (**Gabarito**: Letra D)

(FGV – TRT - 18ª Região (GO) – 2022) Um técnico deseja usar o Keycloak no Docker, instalado e em condições ideais.
I 8080:8080 -e KEYCLOAK_ADMIN=admin -e KEYCLOAK_ADMIN_PASSWORD=admin quay.io/keycloak/keycloak:20.0.0 start-dev

Para iniciar o Keycloak exposto na porta local 8080, criando um usuário inicial admin, com senha admin, a lacuna I deve ser preenchida por:

a) docker run -v
b) docker container start on port
c) docker run -p
d) docker start --port
e) docker container run -d port

Comentários:

A resposta correta é a letra (c). O comando docker run -p é usado para mapear as portas do host para as portas do contêiner. Neste caso, o comando docker run -p 8080:8080 irá mapear a porta 8080 do host para a porta 8080 do contêiner.

As outras opções não são adequadas para este cenário. O comando docker run -v é usado para mapear volumes do host para os volumes do contêiner. O comando docker container start on port não é um comando válido. O comando docker start --port é usado para iniciar um contêiner, mas não mapeia as

portas. O comando docker container run -d port é usado para iniciar um contêiner em modo desconectado, mas não mapeia as portas. (**Gabarito**: Letra C)

CMD

A instrução CMD tem três formas:

- CMD ["executável", "param1", "param2"] (forma exec, esta é a forma preferida).
- CMD ["param1", "param2"] (como parâmetros padrão para o ENTRYPOINT).
- CMD comando param1 param2 (forma shell).

Em um Dockerfile, pode haver apenas uma instrução CMD. Se você listar mais de uma instrução CMD, apenas a última será considerada. O principal objetivo de uma instrução CMD é fornecer padrões para um contêiner em execução. Esses padrões podem incluir um executável, ou podem omitir o executável, nesse caso, você deve especificar uma instrução ENTRYPOINT também.

Se o CMD for usado para fornecer argumentos padrão para a instrução ENTRYPOINT, ambas as instruções CMD e ENTRYPOINT devem ser especificadas no formato de array JSON.

LABEL

A instrução LABEL adiciona metadados a uma imagem. Um LABEL é um par chave-valor. Para incluir espaços em um valor LABEL, use aspas e barras invertidas como faria na análise de linha de comando. Alguns exemplos de uso:

```
LABEL "com.example.vendor"="ACME Incorporated"
LABEL com.example.label-with-value="foo"
LABEL version="1.0"
LABEL description="Este texto ilustra \
que os valores de etiqueta podem se estender por várias linhas."
```

Uma imagem pode ter mais de um rótulo. É possível especificar vários rótulos em uma única linha. Antes do Docker 1.10, isso diminuía o tamanho da imagem final, mas não é mais o caso. Você ainda pode optar por especificar vários rótulos em uma única instrução, de uma das duas maneiras a seguir:

```
LABEL multi.label1="value1" multi.label2="value2" other="value3"

LABEL multi.label1="value1" \
      multi.label2="value2" \
      other="value3"
```

EXPOSE

A instrução EXPOSE informa ao Docker que o contêiner escuta nas portas de rede especificadas em tempo de execução. É possível especificar se a porta escuta TCP ou UDP, e o padrão é TCP se o protocolo não for especificado.

A instrução EXPOSE na verdade não publica a porta. Funciona como uma espécie de documentação entre quem constrói a imagem e quem administra o contêiner, sobre quais portas se pretende publicar. Para realmente publicar a porta ao executar o contêiner, use o sinalizador -p em docker run para publicar e mapear uma ou mais portas, ou o sinalizador -P para publicar todas as portas expostas e mapeá-las para portas de ordem superior.

ENV

A instrução ENV define a variável de ambiente <key> com o valor <value>. Esse valor estará no ambiente para todas as instruções subsequentes no estágio de construção e também poderá ser substituído in-line em muitas delas. O valor será interpretado para outras variáveis de ambiente, portanto, aspas serão removidas se não tiverem escape. Assim como a análise de linha de comando, aspas e barras invertidas podem ser usadas para incluir espaços nos valores.

ADD

O comando ADD tem duas formas:

```
ADD [--chown=<usuário>:<grupo>] [--chmod=<permissões>] [--checksum=<checksum>]
<origem>... <destino>
ADD    [--chown=<usuário>:<grupo>]    [--chmod=<permissões>]    ["<origem>",...
"<destino>"]
```

A segunda forma é necessária quando os caminhos contêm espaços em branco. A instrução ADD copia novos arquivos, diretórios ou URLs de arquivos remotos de <origem> e os adiciona ao sistema de arquivos da imagem no caminho <destino>.

É possível especificar vários recursos <origem>, mas se eles forem arquivos ou diretórios, seus caminhos serão interpretados como relativos ao diretório de origem do contexto da construção.

COPY

A instrução COPY tem duas formas:

COPY [--chown=<usuário>:<grupo>] [--chmod=<permissões>] <origem>... <destino>
COPY [--chown=<usuário>:<grupo>] [--chmod=<permissões>] ["<origem>",... "<destino>"]

A segunda forma é necessária quando os caminhos contêm espaços em branco.

A instrução COPY copia novos arquivos ou diretórios de <origem> e os adiciona ao sistema de arquivos do contêiner no caminho <destino>. É possível especificar vários recursos <origem>, mas os caminhos de arquivos e diretórios serão interpretados como relativos à fonte do contexto da construção.

Além disso, ela oferece várias funcionalidades como a utilização de caracteres curinga, *, para corresponder a múltiplos arquivos baseados em padrões específicos, tornando a inclusão de arquivos mais flexível. Além disso, a instrução COPY requer a especificação do caminho de destino dentro do contêiner, seja um caminho absoluto ou relativo ao diretório de trabalho atual (WORKDIR), facilitando a organização dos arquivos copiados.

Ademais, a instrução COPY permite a definição precisa da propriedade de arquivos e diretórios usando a opção --chown, garantindo permissões adequadas no contêiner. Isso é particularmente importante para manter a segurança e a integridade dos arquivos no ambiente do contêiner.

VOLUME

A instrução VOLUME cria um ponto de montagem com o nome especificado e o aponta como um local que contém volumes montados externamente a partir do hospedeiro nativo ou de outros contêineres. O valor pode ser um array JSON, como VOLUME ["/var/log/"], ou uma string simples com múltiplos argumentos, como VOLUME /var/log ou VOLUME /var/log /var/db.

O comando docker run inicializa o volume recém-criado com quaisquer dados que existam na localização especificada dentro da imagem base. Isso significa que qualquer informação presente nesse ponto de montagem na imagem base será disponibilizada no volume quando um contêiner for iniciado com o comando docker run.

Nos contêineres baseados no sistema operacional Windows, os volumes têm algumas particularidades. O destino de um volume dentro do contêiner deve ser ou um diretório inexistente ou vazio, ou uma unidade diferente de C:. Isso significa que não é possível usar diretamente o diretório C: como ponto de montagem. É importante também ter em mente que qualquer alteração nos dados dentro de um volume após a sua declaração será descartada, o que pode afetar a consistência dos dados. Além disso, ao declarar volumes em um Dockerfile, a formatação deve seguir as regras JSON, com palavras-chave entre aspas duplas, não entre aspas simples.

Por fim, a definição do diretório do hospedeiro (ponto de montagem) ocorre durante a execução do contêiner, não no Dockerfile. Isso é feito para preservar a portabilidade da imagem, uma vez que não é possível garantir que um diretório do hospedeiro estará disponível em todos os sistemas hospedeiros. Portanto, o ponto de montagem do host deve ser especificado ao criar ou executar o contêiner, não no Dockerfile.

WORKDIR

A instrução WORKDIR define o diretório de trabalho para todas as instruções RUN, CMD, ENTRYPOINT, COPY e ADD que a seguem no Dockerfile. Mesmo que o WORKDIR não seja utilizado em nenhuma instrução subsequente do Dockerfile, ele será criado se não existir.

Você pode usar a instrução WORKDIR várias vezes em um Dockerfile. Se um caminho relativo for fornecido, ele será relativo ao caminho do WORKDIR anteriormente definido. Isso permite que você organize o ambiente de trabalho dentro do contêiner de maneira flexível e específica para diferentes etapas da construção da imagem.

ARG

A instrução ARG define uma variável que os usuários podem passar no momento da construção para o construtor usando o comando docker build, utilizando a bandeira --build-arg <nome_da_variável>=<valor>. Se um usuário especificar um argumento de construção que não foi definido no Dockerfile, a construção emitirá um aviso.

```
ARG <name>[=<default value>]
```

Um Dockerfile pode incluir uma ou várias instruções ARG. Por exemplo, o seguinte é um Dockerfile válido:

```
FROM busybox
ARG usuario1
ARG numero_de_construcao
# ...
```

Isso permite aos usuários personalizar a construção da imagem fornecendo valores para essas variáveis durante o processo de construção, tornando a criação de imagens mais flexível e adaptável às necessidades específicas.

(FGV – SEFAZ-AM – 2022) A plataforma Docker pode criar imagens automaticamente, executando as instruções de um arquivo Dockerfile.

A primeira instrução presente em um Dockerfile é denominada

a) ADD.
b) RUN.
c) FROM.
d) EXPOSE.
e) WORKDIR.

Comentários:

A resposta correta é a letra C. A instrução FROM é a primeira instrução que deve ser presente em um Dockerfile. Ela define a imagem base que será usada para construir a imagem.

As outras instruções não são adequadas para esta posição. A instrução ADD é usada para copiar arquivos ou diretórios para a imagem. A instrução RUN é usada para executar comandos na imagem. A instrução EXPOSE é usada para definir as portas que serão expostas pela imagem. A instrução WORKDIR é usada para definir o diretório de trabalho da imagem. (**Gabarito**: Letra C)

Pessoal, apenas sobre "Dockerfile" temos mais de 60 páginas na documentação e mais de 20 questões, ou seja, é um conhecimento que você deve levar para prova!

Arquivo de composição

Agora, imagine que você está construindo uma aplicação composta por vários serviços, cada um executando em seu próprio contêiner Docker. Como você gerencia essa complexidade? É aqui que entra o arquivo de composição.

Um arquivo de composição é um arquivo YAML que define um aplicativo multi-container, é um arquivo que descreve como esses **contêineres interagem entre si.** É usado para criar e executar vários contêineres Docker que trabalham juntos para formar um aplicativo. O arquivo de composição é uma maneira de automatizar o processo de criação e execução de contêineres Docker.

Docker CLI

O Docker executa processos em contêineres isolados. Um contêiner é um processo que é executado em um hospedeiro. O hospedeiro pode ser local ou remoto. Quando um operador executa o comando docker run, o processo do contêiner que é executado é isolado, o que significa que ele possui seu próprio sistema de arquivos, sua própria rede e sua própria árvore de processos isolada, separada do hospedeiro.

O comando básico docker run segue esta forma:

```
$ docker run [OPTIONS] IMAGE[:TAG|@DIGEST] [COMMAND] [ARG...]
```

O comando docker run deve especificar uma IMAGEM da qual derivar o contêiner. Um desenvolvedor de imagem pode definir padrões de imagem relacionados a:

- Execução em segundo plano ou em primeiro plano
- Identificação do contêiner
- Configurações de rede
- Restrições de tempo de execução para CPU e memória

Com o comando docker run [OPTIONS], um operador pode adicionar ou substituir os padrões de imagem definidos por um desenvolvedor. Além disso, os operadores podem substituir quase todos os padrões definidos pelo próprio tempo de execução do Docker. A capacidade do operador de substituir os padrões de imagem e do tempo de execução do Docker é o motivo pelo qual o comando run possui mais opções do que qualquer outro comando do Docker.

Comandos

Com uma ampla gama de comandos disponíveis, os usuários podem realizar diversas operações, desde a criação e execução de contêineres até a gestão de imagens e redes usando o Docker CLI, ou Command Line Interface.

Um dos comandos essenciais é o "docker run", que possibilita a criação de um novo contêiner a partir de uma imagem especificada, permitindo a configuração de diversos parâmetros, como mapeamento de portas, variáveis de ambiente e volumes. Com o "docker build", é possível criar novas imagens Docker com

base em um arquivo chamado Dockerfile, que contém as instruções necessárias para construir a imagem desejada.

Além disso, o comando "docker ps" exibe informações sobre os contêineres em execução, incluindo seus IDs e estados. Vejamos agora uma tabela com os comandos base para o Docker CLI.

Comando	Descrição
attach	Anexa a entrada padrão local, saída e fluxos de erro a um contêiner em execução.
build	Constrói uma imagem a partir de um Dockerfile.
builder	Gerencia construções.
checkpoint	Gerencia checkpoints.
commit	Cria uma nova imagem a partir das alterações de um contêiner.
config	Gerencia configurações do Swarm.
container	Gerencia contêineres.
context	Gerencia contextos.
cp	Copia arquivos/pastas entre um contêiner e o sistema de arquivos local.
create	Cria um novo contêiner.
diff	Inspeciona alterações em arquivos ou diretórios no sistema de arquivos de um contêiner.
events	Obtém eventos em tempo real do servidor.
exec	Executa um comando em um contêiner em execução.
export	Exporta o sistema de arquivos de um contêiner como um arquivo tar.
history	Mostra o histórico de uma imagem.
image	Gerencia imagens.
images	Lista imagens.
import	Importa o conteúdo de um arquivo tarball para criar uma imagem do sistema de arquivos.
info	Exibe informações em todo o sistema.
inspect	Retorna informações de baixo nível sobre objetos Docker.
kill	Encerra um ou mais contêineres em execução.
load	Carrega uma imagem de um arquivo tar ou STDIN.
login	Faz login em um registro (registry).

logout	Faz logout de um registro (registry).
logs	Obtém os registros (logs) de um contêiner.
manifest	Gerencia manifestos e listas de manifestos de imagens Docker.
network	Gerencia redes.
node	Gerencia nós (nodes) do Swarm.
pause	Pausa todos os processos em um ou mais contêineres.
plugin	Gerencia plugins.
port	Lista mapeamentos de porta ou um mapeamento específico para o contêiner.
ps	Lista contêineres.
pull	Faz o download de uma imagem de um registro (registry).
push	Envia uma imagem para um registro (registry).
rename	Renomeia um contêiner.
restart	Reinicia um ou mais contêineres.
rm	Remove um ou mais contêineres.
rmi	Remove uma ou mais imagens.
run	Cria e executa um novo contêiner a partir de uma imagem.
save	Salva uma ou mais imagens em um arquivo tar (transmitido para STDOUT por padrão).
search	Procura imagens no Docker Hub.
secret	Gerencia segredos do Swarm.
service	Gerencia serviços do Swarm.
stack	Gerencia stacks do Swarm.
start	Inicia um ou mais contêineres parados.
stats	Exibe uma transmissão ao vivo das estatísticas de uso de recursos do(s) contêiner(es).
stop	Para um ou mais contêineres em execução.
swarm	Gerencia o Swarm.
system	Gerencia o Docker.
tag	Cria uma tag (rótulo) TARGET_IMAGE que se refere a SOURCE_IMAGE.
top	Exibe os processos em execução de um contêiner.

trust	Gerencia a confiança em imagens Docker.
unpause	Despausa todos os processos em um ou mais contêineres.
update	Atualiza a configuração de um ou mais contêineres.
version	Mostra informações de versão do Docker.
volume	Gerencia volumes.
wait	Bloqueia até que um ou mais contêineres parem e, em seguida, exibe seus códigos de saída.

(CEBRASPE – MPE-RO – 2023) Assinale a opção em que é apresentado o comando que permite listar todos os nós que o gerenciador do Docker Swarm conhece.

a) docker attach
b) docker node ls
c) docker pull
d) docker rmi
e) docker rm /redis

Comentários:

A resposta correta é a letra (b), docker node ls. Este comando lista todos os nós que o gerenciador do Docker Swarm conhece.

As outras opções não são corretas porque:

a) docker attach, é usado para conectar um terminal ao console de um container em execução.
c) docker pull, é usado para baixar uma imagem específica ou um conjunto de imagens do Docker Hub.
d) docker rmi, é usado para remover uma imagem Docker.
e) docker rm /redis, é usado para remover um container.
(**Gabarito**: Letra B)

(CEBRASPE – MPE-RO – 2023) Assinale a opção correspondente ao comando, em Docker, que permite ao usuário fazer o download de uma imagem específica ou um conjunto de imagens do Docker Hub.

a) docker attach
b) docker node ls
c) docker pull
d) docker rmi
e) docker rm / redis

Comentários:

A resposta correta é a letra (c), docker pull. Este comando é usado para baixar uma imagem específica ou um conjunto de imagens do Docker Hub. As outras opções não são corretas porque:

a) docker attach, é usado para conectar um terminal ao console de um container em execução.
b) docker node ls, é usado para listar os nós de um cluster Docker Swarm.
d) docker rmi, é usado para remover uma imagem Docker.
e) docker rm / redis, é usado para remover um container.
(**Gabarito**: Letra C)

(FGV – TJ-DFT – 2022) A analista Sara modificou alguns arquivos do container Docker TJSiteContainer que se encontra em execução. Para criar uma imagem Docker a partir do estado atual de TJSiteContainer, a fim de persistirem as modificações efetuadas em seus arquivos, Sara deve utilizar o comando:

a) docker create
b) docker diff
c) docker export
d) docker image save
e) docker commit

Comentários:

Pessoal, vamos analisar cada opção:

a) docker create: Esse comando é usado para criar um novo container baseado em uma imagem, mas não é relevante para o cenário em questão. Ele não lida com a criação de uma nova imagem a partir das modificações em um container em execução.

b) docker diff: Esse comando é usado para verificar as diferenças entre o sistema de arquivos do container e a imagem base. Ele não cria uma nova imagem a partir das modificações.

c) docker export: Esse comando cria um arquivo tar com o sistema de arquivos do container, mas não inclui informações de imagem ou metadados do Docker. Não é o mais adequado para criar uma nova imagem.

d) docker image save: Esse comando é usado para salvar uma ou mais imagens em um arquivo tar. Ele não lida diretamente com as modificações feitas dentro de um container em execução.

e) docker commit: Este é o comando correto. O comando "docker commit" permite criar uma nova imagem a partir do estado atual de um container. Ele captura o estado atual do sistema de arquivos, os processos em execução e as configurações do container e cria uma nova imagem que reflete essas mudanças. É usado quando você deseja persistir as modificações feitas dentro do container em uma nova imagem.

Portanto, a resposta correta é a letra E) docker commit. Isso permitirá que Sara crie uma nova imagem que contenha as modificações feitas nos arquivos do container TJSiteContainer em execução. (**Gabarito**: Letra E)

(INSTITUTO AOCP – IF-MA – 2023) Osnei é Técnico em Tecnologia da Informação e está trabalhando em um sistema para a instituição em que atua. Ele está utilizando a aplicação de Docker para seu desenvolvimento. Qual comando pode ser utilizado para exibir as versões de API, Client e Server do hots?

a) Docker version.
b) Docker run.
c) Docker api.
d) Docker cliente.
e) Docker server.

Comentários:

A resposta correta é a Letra A. O comando docker version é usado para exibir informações sobre a versão do Docker instalado no sistema. Essas informações incluem as versões da API, do cliente e do servidor do Docker.

As outras alternativas são incorretas porque:

(b): O comando docker run é usado para criar e executar um contêiner Docker.
(c): O comando docker api não existe.
(d): O comando docker cliente não existe.
(e): O comando docker server não existe.

Portanto, a alternativa (a) é o nosso gabarito. (**Gabarito**: Letra A)

docker ps

O comando docker os lista todos os contêineres. Ele exibe informações básicas sobre esses contêineres, como ID, nome, status, portas mapeadas e outros detalhes relevantes. O comando pode ser personalizado usando várias **opções** para atender às necessidades específicas do usuário. Abaixo, vou explicar o uso das **opções mais comuns** disponíveis com o comando "docker ps":

```
docker ps [OPTIONS]
```

Opção	Abreviação	Descrição
--all	-a	Mostra todos os contêineres (por padrão, mostra apenas os em execução).
--filter	-f	Filtra a saída com base nas condições fornecidas.
--format		Formata a saída usando um modelo personalizado: 'table': Imprime a saída em formato de tabela com cabeçalhos de coluna (padrão) 'table TEMPLATE': Imprime a saída em formato de tabela usando o modelo Go fornecido 'json': Imprime em formato JSON 'TEMPLATE':
--last	-n	Mostra os n últimos contêineres criados (inclui todos os estados).
--latest	-l	Mostra o contêiner mais recentemente criado (inclui todos os estados).
--no-trunc		Não trunca a saída.

--quiet	-q	Exibe apenas os IDs dos contêineres.
--size	-s	Exibe os tamanhos totais dos arquivos.

(FGV – MPE-GO – 2022) Um assistente programador está avaliando preventivamente o comportamento de um servidor do MP-GO que executa containers Docker. O assistente deseja verificar os nomes de todos os containers que estão parados e de todos os containers que estão em execução em uma única saída no terminal do servidor.

Para realizar essa verificação usando um simples comando no terminal, o assistente deve usar o comando

a) docker ps -a
b) docker ps -l
c) docker ps -s
d) docker ps -q
e) docker ps

Comentários:

A resposta correta é a letra A. O comando docker ps -a lista todos os containers, incluindo aqueles que estão parados. As outras opções não são adequadas para este cenário. O comando docker ps -l lista apenas o container mais recente. O comando docker ps -s lista o tamanho total dos containers. O comando docker ps -q lista apenas os IDs dos containers. O comando docker ps lista apenas os containers que estão em execução. (**Gabarito**: Letra A)

Docker Compose

O Docker Compose é uma ferramenta que simplifica a orquestração de contêineres Docker para aplicativos compostos por vários serviços. Ele permite definir e executar aplicações multi-conteiner Docker. Com Compose, você **usa um arquivo YAML** para configurar os serviços da sua aplicação. Em seguida, com um único comando, você cria e inicia todos os serviços a partir da sua configuração. Com o Docker Compose, é possível **definir e executar aplicações multi-conteiner** Docker **com um único arquivo**.

Compose funciona criando um gráfico de serviços, que é uma representação visual da sua aplicação. Cada serviço no gráfico é um container Docker que pode ser iniciado e parado individualmente. Para criar um gráfico de serviços, você usa um arquivo YAML. O arquivo YAML contém informações sobre cada serviço, incluindo o nome do serviço, a imagem Docker a ser usada, as portas expostas e as configurações.

O Docker Compose oferece uma série de vantagens uma das principais é a simplificação do processo, permitindo que os desenvolvedores definam e executem aplicações complexas com um único comando, o que acelera o ciclo de desenvolvimento e a criação de ambientes consistentes. Além disso, o Docker Compose é uma ferramenta de código aberto e gratuita, o que o torna acessível a uma ampla gama de desenvolvedores e equipes, promovendo a colaboração e a adoção generalizada da tecnologia de contêineres. Essas vantagens combinadas tornam o Docker Compose uma escolha popular para simplificar a orquestração de aplicativos em contêineres e melhorar a eficiência no desenvolvimento e implantação de software.

Vejamos um exemplo de um arquivo YAML de Compose para uma aplicação web simples:

```
version: "3.7"
services:
  web:
    image: nginx:latest
    ports:
      - "80:80"
  db:
    image: mysql:latest
    ports:
      - "3306:3306"
```

Este arquivo YAML define dois serviços: um servidor web nginx e um servidor de banco de dados MySQL. O servidor web nginx está exposto na porta 80 do host local, e o servidor de banco de dados MySQL está exposto na porta 3306 do host local. Para executar esta aplicação, você pode usar o comando **docker-compose up**. Este comando irá iniciar os dois serviços e disponibilizá-los para uso.

Vejamos agora outro exemplo. Nesse novo exemplo o arquivo YAML define dois serviços: um servidor web nginx e um servidor de banco de dados MySQL. O servidor web nginx está exposto na porta 80 do host local, e o servidor de banco de dados MySQL está exposto na porta 3306 do host local.

```
version: "3.7"
services:
  web:
    image: nginx:latest
    ports:
      - "80:80"
  db:
    image: mysql:latest
    ports:
      - "3306:3306"
```

(FGV – MPE-GO – 2022) Uma equipe de especialistas está implantando uma nova aplicação web que opera em conjunto com uma base de dados. A equipe resolveu utilizar o Docker para esta implantação isolando a aplicação em um container e a base de dados em outro container.

Para definir os dois containers e configurar sua comunicação através de um único arquivo, a equipe deve usar a ferramenta Docker

a) File.
b) Scan.
c) Buildx.
d) Compose.
e) Hub.

Comentários:

A resposta correta é a letra D. A ferramenta Docker Compose é usada para definir e executar multi-container applications. Com o Compose, você usa um único arquivo YAML para definir todos os serviços de sua aplicação. O Compose pode então ser usado para criar, iniciar e parar todos os serviços de sua aplicação com um único comando.

As outras ferramentas não são adequadas para este cenário. O Docker File é usado para construir imagens Docker. O Docker Scan é usado para analisar imagens Docker para vulnerabilidades. O Docker Buildx é usado para construir imagens Docker em paralelo. O Docker Hub é um registro de imagens Docker. (**Gabarito**: Letra D)

(CEFET-MG – CEFET-MG – 2021) A partir do acesso ao terminal de um servidor Debian GNU/Linux com o Docker Engine e Docker Compose devidamente instalados, considere:

- O usuário logado no terminal é o root
- A saída do comando pwd é: /home/user/projeto
- A saída do comando docker image ls é:

```
root@server:/home/user/projeto# docker
image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
```

A saída do comando ls -lha é:

```
root@server:/home/user/projeto# ls -lha
drwxrwxrwx 1 user user 512 Aug 9 10:51 .
drwxrwxrwx 1 user user 512 Aug 9 10:49 ..
-rwxrwxrwx 1 user user 190 Aug 9 10:51 Dockerfile
-rwxrwxrwx 1 user user 121 Aug 9 10:51 docker-compose.yml
```

O conteúdo do arquivo Dockerfile é:

```
FROM debian:stable
RUN apt-get update && apt-get install -y apache2
EXPOSE 80
VOLUME [ “/var/www” , “/var/log/apache2” , “/etc/apache2” ]
ENTRYPOINT [ “/usr/sbin/apache2ctl” , “-D” , “FOREGROUND” ]
```

O conteúdo do arquivo docker-compose.yml é

```
version: "3.9"
services:
  web:
    build: .
    ports:
      - "8080:80"
```

Para se colocar em execução, em segundo plano, um contêiner baseado na imagem especificada pelo arquivo Dockerfile, publicando a porta 8080 do host para a porta 80 do contêiner, é/são suficiente(s) o(s) comando(s):

a) docker build -t imagem-prova:latest . && docker container run -it -p 8080:80 imagem-prova:latest
b) docker build -t imagem-prova:latest . && docker container run -d -p 80:8080 imagem-prova:latest
c) docker container run -d -p 8080:80 imagem-prova:latest
d) docker-compose -f docker-compose.yml up
e) docker-compose up -d

Comentários:

O Docker Compose é uma ferramenta para definir e executar aplicativos Docker com base em arquivos de composição, como o docker-compose.yml. O arquivo docker-compose.yml já contém as configurações necessárias, incluindo a definição do serviço "web" que usa o Dockerfile no diretório atual para construir a imagem. Ao executar docker-compose up -d, você inicia o serviço "web" em segundo plano, seguindo as configurações definidas no arquivo de composição. Isso inclui o mapeamento da porta 8080 do host para a porta 80 do contêiner. Portanto, a opção E é nosso gabarito e irá para colocar em execução o contêiner com base nas configurações fornecidas nos arquivos de composição. Isso simplifica o processo, pois o Docker Compose lida com a construção da imagem e a execução do contêiner com as configurações desejadas. Além disso, as alternativas (A), (B) e (C) não são suficientes para iniciar o contêiner. A alternativa (A) irá construir a imagem imagem-prova:latest e, em seguida, iniciar um contêiner baseado nesta imagem. No entanto, o contêiner não será iniciado em segundo plano. A alternativa (B) irá construir a imagem imagem-prova:latest e, em seguida, iniciar um contêiner baseado nesta imagem, publicando a porta 80 do host para a porta 8080 do contêiner. No entanto, o contêiner não será iniciado em segundo plano. A alternativa (C) irá iniciar um contêiner baseado na imagem imagem-prova:latest, publicando a porta 8080 do host para a porta 80 do contêiner. No entanto, a imagem imagem-prova:latest não existe. (**Gabarito**: Letra E)

Principais tipos de rede Docker

O Docker fornece um subsistema de rede integrado que permite que os contêineres se comuniquem uns com os outros e com o host Docker. O subsistema de rede do Docker é baseado em **drivers**, cada um com suas próprias características e funcionalidades.

Bridge

O Docker utiliza o driver de rede padrão conhecido como "bridge", que cria uma rede virtual para cada contêiner iniciado. Contêineres na mesma rede virtual podem se comunicar através de seus nomes de host. Essa configuração é adequada para a maioria dos contêineres que não necessitam de configurações de rede especiais. Ela é atribuída automaticamente a qualquer contêiner iniciado, a menos que uma rede diferente seja especificamente escolhida. Dentro da rede bridge padrão, a comunicação entre contêineres é possível apenas através de seus endereços IP, a menos que a opção "--link" seja utilizada para estabelecer conexões adicionais, o que é essencial para garantir o isolamento e a segurança dos contêineres no ambiente Docker.

Por outro lado, as redes bridge definidas pelo usuário permitem que contêineres no mesmo host Docker se comuniquem entre si, criando uma rede isolada para grupos de contêineres relacionados a projetos ou

componentes específicos. Em contraste, a rede host compartilha a rede do próprio host com o contêiner, eliminando o isolamento da rede do contêiner em relação ao host.

No âmbito das redes, uma rede bridge é um dispositivo de Camada de Link que encaminha o tráfego entre segmentos de rede. Isso pode ser implementado como um dispositivo de hardware ou como um dispositivo de software que opera no kernel da máquina hospedeira.

Dentro do Docker, uma rede bridge utiliza uma bridge de software para permitir que contêineres conectados à mesma rede bridge se comuniquem, garantindo ao mesmo tempo o isolamento dos contêineres que não estão conectados à mesma rede bridge. O driver de bridge do Docker configura automaticamente regras na máquina hospedeira para impedir que contêineres em redes bridge diferentes se comuniquem diretamente entre si.

As redes bridge se aplicam a contêineres em execução no mesmo host Docker. Para comunicação entre contêineres em hosts Docker diferentes, você pode gerenciar o roteamento no nível do sistema operacional ou optar pelo uso de uma rede de sobreposição. Ao iniciar o Docker, uma rede bridge padrão (também conhecida como "bridge") é criada automaticamente, e os contêineres recém-iniciados se conectam a ela, a menos que seja especificado de outra forma. Além disso, é possível criar redes bridge personalizadas definidas pelo usuário, que são mais flexíveis e adaptáveis às necessidades específicas do projeto.

Host

O driver de rede "host" permite que os contêineres utilizem diretamente a rede do host. Isso significa que o contêiner compartilha a mesma interface de rede que o host Docker e não recebe seu próprio endereço IP. Esse modo de rede é útil quando os contêineres precisam se comunicar com dispositivos na rede local, como servidores web e clientes. Por exemplo, um contêiner destinado a executar um banco de dados pode ser configurado com o driver de rede "host" para permitir a comunicação com clientes na mesma rede. Por outro lado, o driver de rede "overlay" possibilita a comunicação entre contêineres em hosts diferentes e é amplamente utilizado pelo Swarm, a plataforma de orquestração de contêineres do Docker.

O uso do modo de rede "host" implica que o stack de rede do contêiner não está isolado do host Docker, compartilhando o mesmo namespace de rede, e o contêiner não recebe seu próprio endereço IP. Se, por exemplo, você executar um contêiner que está vinculado à porta 80 e usar o modo de rede "host", a aplicação do contêiner estará disponível na porta 80 do endereço IP do host.

O modo de rede "host" pode ser útil para otimizar o desempenho e em cenários em que um contêiner precisa lidar com uma ampla gama de portas, pois não requer tradução de endereços de rede (NAT) e não cria um "proxy de usuário" para cada porta.

É importante notar que o driver de rede "host" funciona apenas em hosts Linux e não é suportado no Docker Desktop para Mac, Docker Desktop para Windows ou Docker EE para Windows Server.

Além disso, é possível utilizar o modo de rede "host" para um serviço de Swarm, passando o parâmetro --network host para o comando docker service create. Nesse caso, o tráfego de controle relacionado à gestão do Swarm e do serviço ainda é enviado através de uma rede de sobreposição, mas os contêineres individuais do serviço de Swarm utilizam a rede e portas do host do Docker daemon, o que impõe algumas limitações,

como permitir apenas um contêiner de serviço em um determinado nó do Swarm, caso um contêiner do serviço se vincule à porta 80.

Macvlan

Para algumas aplicações, especialmente aquelas legadas ou que monitoram o tráfego de rede, é necessário que elas estejam diretamente conectadas à rede física. Nesses casos, o driver de rede macvlan pode ser empregado para atribuir a cada interface de rede virtual do contêiner um endereço MAC exclusivo, fazendo com que ela pareça ser uma interface de rede física diretamente conectada à rede física. Para isso, é necessário definir uma interface física no host do Docker para usar com o Macvlan, bem como a sub-rede e o gateway da rede. É até possível isolar as redes Macvlan usando diferentes interfaces de rede física.

Contudo, é importante considerar o seguinte:

- Há o risco de degradação da rede devido ao esgotamento de endereços IP ou à disseminação excessiva de VLAN, que ocorre quando há um número excessivo de endereços MAC únicos na rede.
- Seu equipamento de rede deve ser capaz de suportar o "modo promíscuo", onde uma única interface física pode ser atribuída a múltiplos endereços MAC.
- Se a sua aplicação puder funcionar com uma rede bridge (em um único host Docker) ou overlay (para comunicação entre múltiplos hosts Docker), essas soluções podem ser mais vantajosas a longo prazo.

Por outro lado, o driver de rede macvlan é uma alternativa para contêineres que necessitam interagir com dispositivos que esperam se comunicar com elementos físicos.

O Docker também oferece suporte para a instalação e utilização de plugins de rede de terceiros, que podem ser empregados para acrescentar novos recursos e funcionalidades ao sistema de rede do Docker.

Overlay

As redes Overlay são mais adequadas quando há a necessidade de comunicação entre contêineres em execução em diferentes hosts Docker ou quando diversos aplicativos colaboram usando os serviços do Swarm.

O driver de rede "overlay" cria uma rede distribuída entre vários hosts do Docker. Essa rede é implementada sobre as redes específicas de cada host (overlay), permitindo que contêineres conectados a ela (incluindo contêineres de serviços de swarm) se comuniquem de forma segura quando a criptografia está habilitada. O Docker gerencia de forma transparente o roteamento de cada pacote de e para o host Docker correto e o contêiner de destino correto.

Quando você inicializa um swarm ou conecta um host Docker a um swarm existente, dois novos tipos de redes são criados no host do Docker:

- Uma rede overlay chamada "ingress", que lida com o tráfego de controle e de dados relacionado aos serviços do swarm. Quando você cria um serviço de swarm e não o conecta a uma rede overlay definida pelo usuário, ele se conecta à rede "ingress" por padrão.

- Uma rede bridge chamada "docker_gwbridge", que conecta o daemon Docker individual aos outros daemons que participam do swarm.

Você pode criar redes overlay definidas pelo usuário usando o comando "docker network create", da mesma forma que cria redes bridge definidas pelo usuário. Serviços ou contêineres podem estar conectados a mais de uma rede ao mesmo tempo, mas só podem se comunicar através das redes às quais cada um está conectado. Embora você possa conectar tanto serviços de swarm quanto contêineres autônomos a uma rede overlay, os comportamentos padrão e as preocupações de configuração são diferentes. Por essa razão, o restante deste tópico está dividido em operações que se aplicam a todas as redes overlay, aquelas que se aplicam a redes de serviços de swarm e aquelas que se aplicam a redes overlay usadas por contêineres autônomos.

Macvlan

As redes Macvlan são a escolha ideal ao migrar de uma configuração de máquina virtual ou quando é preciso que os contêineres se assemelhem a hosts físicos na rede, cada um com seu próprio endereço MAC exclusivo.

Quando há restrições no número de endereços MAC que podem ser atribuídos a uma interface ou porta de rede, considera-se o uso do IPvlan, que é semelhante ao Macvlan, mas não atribui endereços MAC exclusivos aos contêineres. Por fim, a integração do Docker com pilhas de rede especializadas é possível por meio de plugins de rede de terceiros.

Algumas aplicações, especialmente as legadas ou aquelas que monitoram o tráfego de rede, esperam estar diretamente conectadas à rede física. Nesse tipo de situação, você pode utilizar o driver de rede macvlan para atribuir um endereço MAC a cada interface de rede virtual do contêiner, fazendo com que ela pareça ser uma interface de rede física diretamente conectada à rede física. Nesse caso, é necessário designar uma interface física no seu host do Docker para usar com o Macvlan, bem como definir a sub-rede e o gateway da rede. Você até mesmo pode isolar suas redes Macvlan utilizando diferentes interfaces de rede física. Algumas considerações importantes:

- Você pode inadvertidamente prejudicar sua rede devido ao esgotamento de endereços IP ou à "disseminação de VLAN", uma situação que ocorre quando você tem um número inapropriadamente grande de endereços MAC exclusivos em sua rede.

- Seus equipamentos de rede precisam ser capazes de lidar com o "modo promíscuo", onde uma única interface física pode ser atribuída a múltiplos endereços MAC.

- Se a sua aplicação puder funcionar usando uma rede bridge (em um único host Docker) ou overlay (para comunicação entre vários hosts Docker), essas soluções podem ser mais adequadas a longo prazo.

None

Já, o driver de rede none permite que os contêineres sejam completamente isolados da rede. A rede None é uma rede virtual que não fornece nenhuma interface para o container. O container é completamente isolado da rede, incluindo do host Docker. Ela é útil para containers que não precisam se comunicar com

outros dispositivos. Por exemplo, um container que é usado para executar um processo de backup não precisa se comunicar com outros dispositivos. Portanto, pode ser executado em uma rede None.

Vejamos o compilado das informações de cada uma das redes.

1. Bridge Network (Rede Bridge): É a rede padrão do Docker. Ela cria uma rede interna isolada onde os contêineres podem se comunicar entre si. Cada contêiner recebe um endereço IP exclusivo dentro dessa rede e pode ser acessado por outros contêineres usando esse endereço.
2. Host Network (Rede Host): Nesse tipo de rede, os contêineres compartilham a mesma interface de rede do host. Isso significa que eles não têm isolamento de rede e podem usar as portas do host diretamente. Os contêineres podem se comunicar com outros serviços em execução no host, mas não há isolamento de rede entre eles.
3. Overlay Network (Rede Overlay): A rede de sobreposição é usada para conectar contêineres em diferentes hosts Docker em um ambiente distribuído. Ela permite a comunicação entre contêineres em hosts diferentes, como em um cluster Docker Swarm.
4. Macvlan Network (Rede Macvlan): Esse tipo de rede permite atribuir endereços MAC e IP reais aos contêineres, fazendo com que pareçam dispositivos físicos conectados à rede física. Isso é útil quando você precisa que os contêineres tenham conectividade direta com a rede física.
5. None Network (Rede None): Essa rede **desativa a interface de rede** do contêiner, **tornando-o isolado e sem conectividade de rede**. É útil quando você **não deseja que o contêiner tenha acesso à rede**.

Para visualizar todas as redes, incluindo as redes que não estão em execução, deve-se executar o comando **docker network ls.**

(FCC – TJ-CE – 2022) No Linux, em condições ideais, o Docker é disponibilizado com três redes por padrão. Essas redes oferecem configurações específicas para o gerenciamento do tráfego de dados e

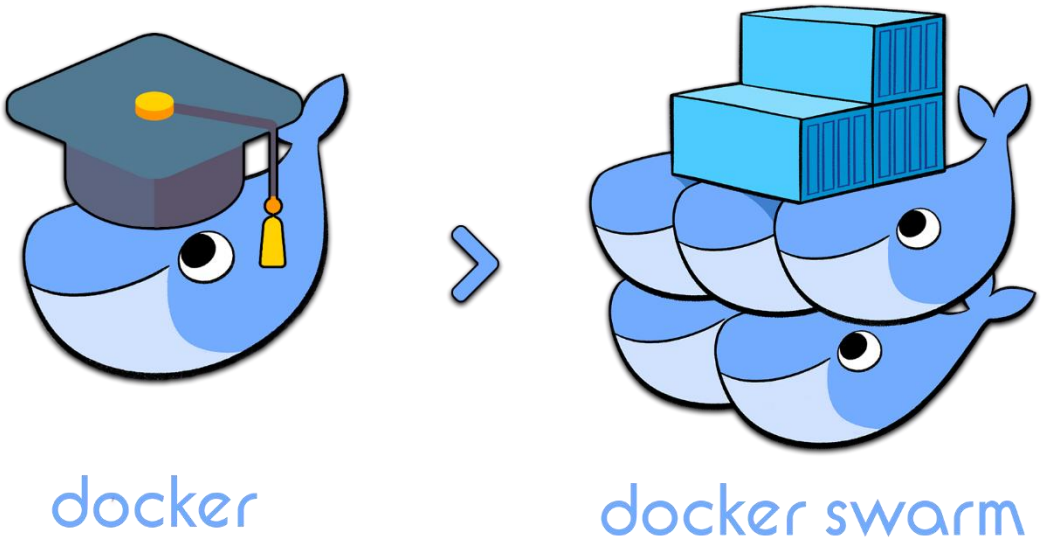
a) a rede None tem como objetivo entregar para o container todas as interfaces existentes no docker none. Isso agiliza a entrega dos pacotes, uma vez que não há host no caminho das mensagens. Mas o uso de um host pode ser importante para a segurança e a gerência do tráfego.
b) Bridge é a rede padrão para qualquer container iniciado no Docker, a menos que seja, explicitamente, associada outra rede a ele. Os containers da rede default bridge podem acessar uns aos outros somente através de seus endereços IP, a menos que se utilize a opção --link
c) deve-se, para visualizar as redes no Docker, utilizar o comando: docker network -show
d) todos os containers que estão na rede None poderão se comunicar via protocolo TCP/IP. Se uma pessoa souber qual é o endereço IP do container que deseja conectar, é possível enviar tráfego para ele, pois estão todos na mesma rede IP (172.17.0.0/24).
e) a rede Host tem como objetivo isolar o container para comunicações externas. A rede não recebe qualquer interface para comunicação externa. A única interface de rede IP será a localhost. Essa rede, normalmente, é utilizada para containers que manipulam apenas arquivos de backup, sem necessidade de enviá-los via rede para outro local.

Comentários:

Pessoal, a resposta correta é a letra B. As redes padrão do Docker no Linux são Bridge, Host e None. A rede Bridge é a rede padrão para qualquer container iniciado no Docker, a menos que seja, explicitamente,

associada outra rede a ele. Os containers da rede default bridge podem acessar uns aos outros somente através de seus endereços IP, a menos que se utilize a opção --link. Por outro lado, a rede Host é uma rede que permite que um container compartilhe a rede do host. Os containers da rede host podem acessar o host diretamente e outros containers na rede host. Por fim, a rede None: É uma rede que não permite que um container se conecte a nenhuma outra rede. Os containers da rede none não podem se comunicar com outros containers ou com o host. Portanto, nosso gabarito é a Letra B. (**Gabarito**: Letra B)

Docker Swarm



Docker Swarm é uma ferramenta de **orquestração de containers** que permite **gerenciar grupos de containers Docker**. Ele é útil para **gerenciar aplicativos complexos** que são compostos de vários containers.

É usado para simplificar a implantação, o escalonamento e a alta disponibilidade de aplicativos em contêineres em ambientes de produção. O termo "Swarm" (em português, "**enxame**") reflete a ideia de que os **contêineres em um cluster Docker Swarm trabalham juntos** como uma coleção coordenada, semelhante a uma **colmeia de abelhas**.

Além disso, o Docker Swarm ajusta automaticamente o número de contêineres em execução para **atender às demandas de tráfego**, garantindo alta disponibilidade e escalabilidade sob demanda. A ferramenta também fornece recursos de monitoramento para acompanhar a saúde dos contêineres e do cluster.

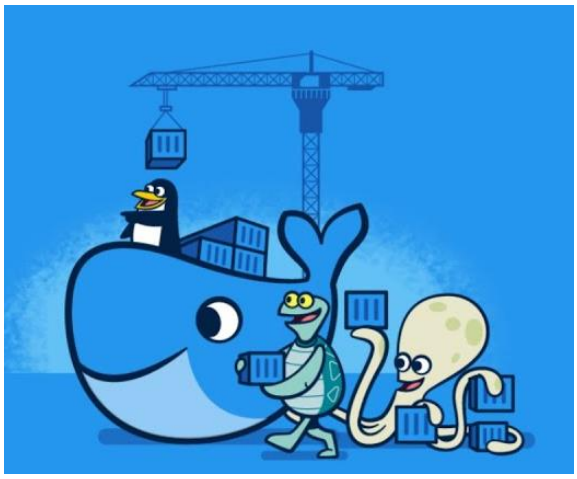
O Docker Swarm oferece um gerenciamento de cluster integrado ao **Docker Engine**, eliminando a necessidade de software de orquestração adicional para criar e administrar um enxame de Docker Engines. Sua abordagem descentralizada permite que os Docker Engines lidem com especializações em tempo de execução, permitindo que você construa um enxame inteiro com base em uma única imagem de disco. Além disso, ele adota um modelo de serviço declarativo, onde você pode definir o estado desejado dos serviços em sua pilha de aplicativos.

A escalabilidade é uma característica fundamental do Docker Swarm, pois permite que você declare o número de tarefas desejadas para cada serviço e o gerenciador de enxame se adapta automaticamente,

adicionando ou removendo tarefas para manter o estado desejado. O Swarm também oferece reconciliação do estado desejado, monitorando constantemente o estado do cluster e resolvendo quaisquer diferenças entre o estado real e o estado desejado.

Além disso, o Docker Swarm fornece uma rede multi-host com sobreposição, descoberta de serviço, balanceamento de carga e segurança por padrão. Ele permite expor as portas dos serviços a um balanceador de carga externo, facilitando a exposição dos serviços para acesso externo. A segurança é enfatizada com **autenticação mútua e criptografia TLS** entre os nós swarm, garantindo a proteção das comunicações. Além disso, o Docker Swarm permite atualizações contínuas de serviços, oferecendo controle sobre o processo de implantação e a capacidade de reverter para versões anteriores em caso de problemas.

Docker Desktop



O Docker Desktop é um aplicativo de instalação simplificada, com apenas um clique, desenvolvido para ambientes Mac, Linux e Windows. Ele tem como objetivo facilitar o processo de construção, compartilhamento e execução de aplicativos e microsserviços em contêineres dentro do seu ambiente de desenvolvimento local.

Esta ferramenta oferece uma interface gráfica de usuário (GUI) intuitiva que permite aos usuários gerenciar contêineres, aplicativos e imagens de forma direta a partir de suas máquinas. O Docker Desktop pode ser utilizado isoladamente ou como uma ferramenta complementar à Interface de Linha de Comando (CLI) do Docker.

Uma das principais vantagens do Docker Desktop é sua capacidade de **reduzir o tempo e o esforço** necessários **para configurações complexas**. Ele cuida de tarefas como mapeamento de portas, configurações do sistema de arquivos e outras configurações padrão, permitindo que os desenvolvedores se concentrem na escrita de código. Além disso, o Docker Desktop recebe atualizações regulares que incluem correções de bugs e melhorias de segurança, garantindo um ambiente de desenvolvimento estável e seguro.

REFERÊNCIAS

Documentação oficial Docker – docs.docker.com

Docker – https://pt.wikiversity.org/wiki/Docker

QUESTÕES COMENTADAS

1. **(CEBRASPE – MPE-RO – 2023)** Assinale a opção correspondente ao comando, em Docker, que permite ao usuário fazer o download de uma imagem específica ou um conjunto de imagens do Docker Hub.

a) docker attach
b) docker node ls
c) docker pull
d) docker rmi
e) docker rm / redis

Comentários:

A resposta correta é a letra (c), docker pull. Este comando é usado para baixar uma imagem específica ou um conjunto de imagens do Docker Hub. As outras opções não são corretas porque:

a) docker attach, é usado para conectar um terminal ao console de um container em execução.
b) docker node ls, é usado para listar os nós de um cluster Docker Swarm.
d) docker rmi, é usado para remover uma imagem Docker.
e) docker rm / redis, é usado para remover um container.

Gabarito: Letra C

---

2. **(CEBRASPE – MPE-RO – 2023)** Assinale a opção em que é apresentado o comando que permite listar todos os nós que o gerenciador do Docker Swarm conhece.

a) docker attach
b) docker node ls
c) docker pull
d) docker rmi
e) docker rm /redis

Comentários:

A resposta correta é a letra (b), docker node ls. Este comando lista todos os nós que o gerenciador do Docker Swarm conhece.

As outras opções não são corretas porque:

a) docker attach, é usado para conectar um terminal ao console de um container em execução.
c) docker pull, é usado para baixar uma imagem específica ou um conjunto de imagens do Docker Hub.
d) docker rmi, é usado para remover uma imagem Docker.
e) docker rm /redis, é usado para remover um container.

Gabarito: Letra B

---

3. **(CEBRASPE – SEFIN de Fortaleza - CE – 2023)** Tendo em vista que, no atual cenário de desenvolvimento de aplicações web, é essencial considerar princípios, como consistência e escalabilidade, e práticas, como automação do processo de implantação e integração do código-fonte, julgue o item subsequente.

   Um exemplo prático de conteinerização de aplicação é a utilização do Docker para criar um ambiente consistente; nesse caso, é correto criar um arquivo Dockerfile por meio do comando docker build.

Comentários:

A afirmação é errada. O comando docker build é usado para construir uma imagem Docker a partir de um arquivo Dockerfile. O Dockerfile é um arquivo de texto que contém instruções sobre como construir uma imagem Docker.

Portanto, criar um arquivo Dockerfile não é um exemplo prático de conteinerização de aplicação. Na verdade, é uma etapa necessária para criar uma imagem Docker, que é um componente essencial da conteinerização de aplicações.

Um exemplo prático de conteinerização de aplicação é criar um container Docker a partir de uma imagem Docker. Para fazer isso, usamos o comando docker run.

Gabarito: Errado

---

4. **(FGV – TJ-RN – 2023)** A analista Maria gerencia a aplicação WebJus. A WebJus requer a execução de dois containers Docker que necessitam de comunicação entre si. No entanto, para fins de balanceamento de carga, cada container da WebJus foi alocado em um servidor diferente. A fim de configurar a comunicação entre os containers da WebJus de forma simples, Maria criou a rede Docker NetJus, que é capaz de conectar containers rodando em servidores diferentes.

   A NetJus utiliza o driver de rede Docker:

   a) host;
   b) ipvlan;
   c) bridge;
   d) overlay;
   e) macvlan.

Comentários:

No cenário descrito, em que Maria gerencia a aplicação WebJus utilizando containers Docker que precisam de comunicação entre si, mas estão alocados em servidores diferentes para balanceamento de carga, a opção mais apropriada para criar a rede que permite a comunicação entre esses containers é usando o driver de rede "overlay".

O driver de rede "overlay" permite criar redes virtuais que podem se estender além dos limites de um único host. Isso é especialmente útil quando você precisa conectar containers em diferentes hosts para que eles

possam se comunicar como se estivessem na mesma rede local. O driver "overlay" cria uma rede sobreposta que utiliza túneis para transportar o tráfego entre os hosts, permitindo que os containers vejam uns aos outros como se estivessem na mesma rede, independentemente de sua localização física.

Portanto, a utilização do driver "overlay" é a solução apropriada para o caso apresentado, permitindo que os containers da WebJus, alocados em servidores diferentes, possam se comunicar de forma eficiente através da rede Docker NetJu

Gabarito: Letra D

---

5. **(UFPR – IF-PR – 2023)** O comando docker run cria e executa um novo contêiner a partir de uma imagem. Para a criação desse contêiner em segundo plano, utiliza-se o comando:

a) docker container run -b
b) docker container run -p
c) docker container run -a
d) docker container run -d
e) docker container run -s

Comentários:

O comando "docker run" é usado para criar e executar um novo contêiner a partir de uma imagem. Para executar o contêiner em segundo plano, ou seja, em modo daemon (background), utiliza-se a opção "-d". Portanto, a resposta correta é a opção "d) docker container run -d". Isso permite que o contêiner seja executado em segundo plano, liberando o terminal para que você possa continuar a usar a linha de comando para outras tarefas.

Gabarito: Letra D

---

6. **(INSTITUTO AOCP – IF-MA – 2023)** Osnei é Técnico em Tecnologia da Informação e está trabalhando em um sistema para a instituição em que atua. Ele está utilizando a aplicação de Docker para seu desenvolvimento. Qual comando pode ser utilizado para exibir as versões de API, Client e Server do hots?

a) Docker version.
b) Docker run.
c) Docker api.
d) Docker cliente.
e) Docker server.

Comentários:

A resposta correta é a Letra A. O comando docker version é usado para exibir informações sobre a versão do Docker instalado no sistema. Essas informações incluem as versões da API, do cliente e do servidor do Docker.

As outras alternativas são incorretas porque:

(b): O comando docker run é usado para criar e executar um contêiner Docker.
(c): O comando docker api não existe.
(d): O comando docker cliente não existe.
(e): O comando docker server não existe.

Portanto, a alternativa (a) é o nosso gabarito.

Gabarito: Letra A

---

7. **(UFPR – IF-PR – 2023)** O comando docker run cria e executa um novo contêiner a partir de uma imagem. Para a criação desse contêiner em segundo plano, utiliza-se o comando:

a) docker container run -b
b) docker container run -p
c) docker container run -a
d) docker container run -d
e) docker container run -s

Comentários:

O comando "docker run" é usado para criar e executar um novo contêiner a partir de uma imagem. Para executar o contêiner em segundo plano, ou seja, em modo daemon (background), utiliza-se a opção "-d". Portanto, a resposta correta é a opção "d) docker container run -d". Isso permite que o contêiner seja executado em segundo plano, liberando o terminal para que você possa continuar a usar a linha de comando para outras tarefas.

Gabarito: Letra D

---

8. **(FGV – SEFAZ-AM – 2022)** Leia o fragmento a seguir.
"A plataforma Docker usa uma arquitetura do tipo _____. O cliente Docker conversa com o daemon do Docker, que constrói, executa e distribui _____ Docker. O cliente e o daemon do Docker podem ser executados em um mesmo sistema ou se conectar um cliente do Docker a um daemon remoto. O cliente Docker e o daemon se comunicam usando _____ ou uma interface de redes."

Assinale a opção cujos itens completam corretamente as lacunas do fragmento acima, na ordem apresentada.

a) MVC – imagens – chamadas RPC ou bluetooth.
b) thin client – contêineres – wireless ou bluetooth.
c) serverless – componentes – chamadas MPI ou RPC.
d) cliente-servidor – contêineres – API REST ou soquetes UNIX.
e) mesh app and service – imagens – API RESTFULL ou wireless.

Comentários:

Pessoal, o fragmento menciona o funcionamento da plataforma Docker e descreve como os diferentes componentes interagem. Vamos analisar cada parte do fragmento:

"A plataforma Docker usa uma arquitetura do tipo _____."
Aqui, o texto se refere à arquitetura subjacente que o Docker utiliza. O Docker segue uma arquitetura cliente-servidor. "O cliente Docker conversa com o daemon do Docker, que constrói, executa e distribui _____ Docker." O cliente Docker é a interface por meio da qual os usuários interagem com o Docker. Ele se comunica com o daemon Docker, que é o processo responsável por gerenciar os containers. O daemon constrói, executa e distribui imagens Docker.

"O cliente e o daemon do Docker podem ser executados em um mesmo sistema ou se conectar a um cliente do Docker a um daemon remoto." Isso significa que tanto o cliente quanto o daemon podem estar no mesmo sistema (máquina) ou podem estar em sistemas diferentes, permitindo a conexão a um daemon remoto, se necessário.

"O cliente Docker e o daemon se comunicam usando _____ ou uma interface de redes." Aqui, o texto aborda os métodos de comunicação entre o cliente Docker e o daemon Docker. Eles se comunicam usando uma API REST (Representational State Transfer) ou soquetes UNIX.

A opção correta que preenche corretamente as lacunas do fragmento é:

d) cliente-servidor - contêineres - API REST ou soquetes UNIX.

Portanto, o Docker opera em uma arquitetura cliente-servidor, onde o cliente Docker se comunica com o daemon Docker. O daemon constrói, executa e distribui contêineres Docker. A comunicação entre o cliente e o daemon é realizada por meio de uma API REST ou soquetes UNIX, dependendo do cenário.

Gabarito: Letra D

---

9. **(FGV – TJ-DFT – 2022)** A analista Sara modificou alguns arquivos do container Docker TJSiteContainer que se encontra em execução. Para criar uma imagem Docker a partir do estado atual de TJSiteContainer, a fim de persistirem as modificações efetuadas em seus arquivos, Sara deve utilizar o comando:

a) docker create
b) docker diff
c) docker export
d) docker image save
e) docker commit

Comentários:

Pessoal, vamos analisar cada opção:

a) docker create: Esse comando é usado para criar um novo container baseado em uma imagem, mas não é relevante para o cenário em questão. Ele não lida com a criação de uma nova imagem a partir das modificações em um container em execução.

b) docker diff: Esse comando é usado para verificar as diferenças entre o sistema de arquivos do container e a imagem base. Ele não cria uma nova imagem a partir das modificações.

c) docker export: Esse comando cria um arquivo tar com o sistema de arquivos do container, mas não inclui informações de imagem ou metadados do Docker. Não é o mais adequado para criar uma nova imagem.

d) docker image save: Esse comando é usado para salvar uma ou mais imagens em um arquivo tar. Ele não lida diretamente com as modificações feitas dentro de um container em execução.

e) docker commit: Este é o comando correto. O comando "docker commit" permite criar uma nova imagem a partir do estado atual de um container. Ele captura o estado atual do sistema de arquivos, os processos em execução e as configurações do container e cria uma nova imagem que reflete essas mudanças. É usado quando você deseja persistir as modificações feitas dentro do container em uma nova imagem.

Portanto, a resposta correta é a letra E) docker commit. Isso permitirá que Sara crie uma nova imagem que contenha as modificações feitas nos arquivos do container TJSiteContainer em execução.

Gabarito: Letra E

---

10. (FGV – SEFAZ-AM – 2022) Um Docker Hub consiste em um índice e registro do Docker. Com relação ao registro do Docker, analise as afirmativas a seguir e assinale (V) para a verdadeira e (F) para a falsa.

( ) Possui recursos avançados que incluem bugsnag, new relic e cors.

( ) Suporta diferentes tipos de back-end de armazenamento de arquivos em nuvem ou sistema de arquivos local.

( ) Armazena dados referentes às contas de usuários em banco de dados local.

As afirmativas são, na ordem apresentada, respectivamente

a) F - V - V.
b) V - V - F.
c) F - F - V.
d) F - V - F.
e) V - F - F.

Comentários:

A afirmativa 1 é verdadeira, pois o Docker Hub oferece uma variedade de recursos avançados, incluindo bugsnag, new relic e cors. A afirmativa 2 é verdadeira, pois o Docker Hub suporta diferentes tipos de back-end de armazenamento, incluindo armazenamento de arquivos em nuvem e sistema de arquivos local. A afirmativa 3 é falsa, pois os dados referentes às contas de usuários são armazenados em banco de dados MongoDB, que não é um banco de dados local. O MongoDB é um banco de dados de documentos NoSQL que pode ser executado em servidores físicos ou virtuais.

Gabarito: Letra B

11. (FGV – CGU – 2022) Uma das estratégias para reduzir o tamanho de imagens Docker consiste em:

a) combinar comandos RUN em um único comando;
b) substituir a imagem base por uma versão mais recente;
c) separar comandos RUN complexos em comandos menores;
d) reordenar os comandos de forma que o cache seja utilizado com maior frequência;
e) compactar arquivos a serem copiados para a imagem e descompactá-los durante a sua geração

Comentários:

Pessoal, nessa questão é importante entender o que ocorre em cada caso específico[]
(a) Combinar comandos RUN em um único comando

Ao combinar comandos RUN em um único comando, você evita que os arquivos intermediários sejam criados. Isso ocorre porque os comandos RUN são executados sequencialmente, e os arquivos intermediários são criados para cada comando. Ao combinar comandos RUN, você pode executar todos os comandos de uma vez, o que evita que os arquivos intermediários sejam criados.

(b) Substituir a imagem base por uma versão mais recente

Imagens mais recentes geralmente são menores do que versões anteriores. Isso ocorre porque as imagens mais recentes são atualizadas com as últimas correções e otimizações.

Por exemplo, suponha que você esteja usando uma imagem base de Ubuntu 18.04. Essa imagem é relativamente grande, com cerca de 2 GB. Você pode reduzir o tamanho da imagem substituindo a imagem base por uma versão mais recente, como o Ubuntu 20.04. Essa imagem é menor, com cerca de 1,5 GB.

(c) Separar comandos RUN complexos em comandos menores

Comandos RUN complexos podem criar arquivos intermediários grandes. Separando esses comandos em comandos menores, você pode reduzir o tamanho dos arquivos intermediários.

(d) Reordenar os comandos de forma que o cache seja utilizado com maior frequência

O cache do Docker pode ajudar a reduzir o tempo de construção da imagem, mas também pode aumentar o tamanho da imagem. Reordenando os comandos de forma que o cache seja utilizado com maior frequência, você pode reduzir o tamanho da imagem.

(e) Compactar arquivos a serem copiados para a imagem e descompactá-los durante a sua geração

Arquivos compactados são menores do que arquivos não compactados. Compactando arquivos a serem copiados para a imagem, você pode reduzir o tamanho da imagem.

Gabarito: Letra A

12. (FGV – MPE-GO – 2022) Uma equipe de especialistas está implantando uma nova aplicação web que opera em conjunto com uma base de dados. A equipe resolveu utilizar o Docker para esta implantação isolando a aplicação em um container e a base de dados em outro container.

Para definir os dois containers e configurar sua comunicação através de um único arquivo, a equipe deve usar a ferramenta Docker

a) File.
b) Scan.
c) Buildx.
d) Compose.
e) Hub.

Comentários:

A resposta correta é a letra D. A ferramenta Docker Compose é usada para definir e executar multi-container applications. Com o Compose, você usa um único arquivo YAML para definir todos os serviços de sua aplicação. O Compose pode então ser usado para criar, iniciar e parar todos os serviços de sua aplicação com um único comando.

As outras ferramentas não são adequadas para este cenário. O Docker File é usado para construir imagens Docker. O Docker Scan é usado para analisar imagens Docker para vulnerabilidades. O Docker Buildx é usado para construir imagens Docker em paralelo. O Docker Hub é um registro de imagens Docker.

Gabarito: Letra D

---

13. (FGV – SEFAZ-AM – 2022) A plataforma Docker pode criar imagens automaticamente, executando as instruções de um arquivo Dockerfile.

A primeira instrução presente em um Dockerfile é denominada

a) ADD.
b) RUN.
c) FROM.
d) EXPOSE.
e) WORKDIR.

Comentários:

A resposta correta é a letra C. A instrução FROM é a primeira instrução que deve ser presente em um Dockerfile. Ela define a imagem base que será usada para construir a imagem.

As outras instruções não são adequadas para esta posição. A instrução ADD é usada para copiar arquivos ou diretórios para a imagem. A instrução RUN é usada para executar comandos na imagem. A instrução EXPOSE é usada para definir as portas que serão expostas pela imagem. A instrução WORKDIR é usada para definir o diretório de trabalho da imagem.

Gabarito: Letra C

---

14. (FGV – TRT - 16ª REGIÃO (MA) – 2022) Docker é uma plataforma que permite criar e compartilhar aplicativos e microsserviços em contêineres.

O comando utilizado para executar um contêiner novo é o

a) docker composes.
b) docker create.
c) docker build.
d) docker exec.
e) docker run.

Comentários:

A resposta correta é a letra E. O comando docker run é usado para executar um contêiner novo. Ele pode ser usado para iniciar um contêiner a partir de uma imagem existente ou para construir e iniciar um contêiner a partir de um Dockerfile.

As outras opções não são adequadas para este cenário. O comando docker compose é usado para iniciar um conjunto de contêineres. O comando docker create é usado para criar um contêiner, mas não o inicia. O comando docker build é usado para construir uma imagem Docker. O comando docker exec é usado para executar um comando em um contêiner já em execução.

Gabarito: Letra E

---

15. (FGV – MPE-GO – 2022) Um assistente programador está avaliando preventivamente o comportamento de um servidor do MP-GO que executa containers Docker. O assistente deseja verificar os nomes de todos os containers que estão parados e de todos os containers que estão em execução em uma única saída no terminal do servidor.

Para realizar essa verificação usando um simples comando no terminal, o assistente deve usar o comando

a) docker ps -a
b) docker ps -l
c) docker ps -s
d) docker ps -q
e) docker ps

Comentários:

A resposta correta é a letra A. O comando docker ps -a lista todos os containers, incluindo aqueles que estão parados. As outras opções não são adequadas para este cenário. O comando docker ps -l lista apenas o container mais recente. O comando docker ps -s lista o tamanho total dos containers. O comando docker

ps -q lista apenas os IDs dos containers. O comando docker ps lista apenas os containers que estão em execução.

Gabarito: Letra A

---

16. (FGV – SEFAZ-AM – 2022) Com relação à segurança dos contêineres Docker, analise as afirmativas a seguir e assinale (V) para a verdadeira e (F) para a falsa.

( ) Possuem isolamento no nível do processo no sistema operacional e isolamento adicional, usando recursos especiais tais como namespaces e cgroups.

( ) Aproveitam-se dos mecanismos interprocess communication padrão, tais como sinais, pipes e sockets, onde cada contêiner possui a sua própria network stack.

( ) Os contêineres possuem um multi level security no sistema operacional do host e os recursos físicos que são gerenciados por um hypervisor compartilhado.

As afirmativas são, na ordem apresentada, respectivamente,

a) F - V - V.
b) F - F - V.
c) V - F - V.
d) V - F - F.
e) V - V - F.

Comentários:

A primeira afirmativa é verdadeira. Os contêineres Docker são isolados do sistema operacional host no nível do processo. Isso significa que eles não compartilham recursos do sistema operacional, como memória, CPU e armazenamento.

A segunda afirmativa também é verdadeira. Os contêineres Docker aproveitam-se dos mecanismos de comunicação interprocesso padrão, tais como sinais, pipes e sockets. Isso permite que os contêineres se comuniquem entre si e com o sistema operacional host.

A terceira afirmativa é falsa. Os contêineres Docker não possuem um multi level security no sistema operacional do host. Eles são isolados do sistema operacional host, mas ainda compartilham o mesmo nível de segurança. Assim, nosso gabarito é a letra E!

Gabarito: Letra E

---

17. (FGV – TRT - 18ª Região (GO) – 2022) Um técnico deseja usar o Keycloak no Docker, instalado e em condições ideais.
I 8080:8080 -e KEYCLOAK_ADMIN=admin -e KEYCLOAK_ADMIN_PASSWORD=admin quay.io/keycloak/keycloak:20.0.0 start-dev

Para iniciar o Keycloak exposto na porta local 8080, criando um usuário inicial admin, com senha admin, a lacuna I deve ser preenchida por:

a) docker run -v
b) docker container start on port
c) docker run -p
d) docker start --port
e) docker container run -d port

Comentários:

A resposta correta é a letra (c). O comando docker run -p é usado para mapear as portas do host para as portas do contêiner. Neste caso, o comando docker run -p 8080:8080 irá mapear a porta 8080 do host para a porta 8080 do contêiner.

As outras opções não são adequadas para este cenário. O comando docker run -v é usado para mapear volumes do host para os volumes do contêiner. O comando docker container start on port não é um comando válido. O comando docker start --port é usado para iniciar um contêiner, mas não mapeia as portas. O comando docker container run -d port é usado para iniciar um contêiner em modo desconectado, mas não mapeia as portas.

Gabarito: Letra C

---

18. (FCC – TRT - 23ª REGIÃO (MT) – 2022) Um Técnico precisa administrar e monitorar containers Docker em produção. Para isso pode utilizar a ferramenta

a) Rancher
b) Webhooker
c) Swagger
d) Zuul
e) Flyway

Comentários:

A resposta correta é a letra (a). Rancher é uma plataforma de gerenciamento de contêineres de código aberto que pode ser usada para administrar e monitorar containers Docker em produção. Rancher fornece uma interface gráfica de usuário (GUI) e uma API para gerenciar containers Docker. Rancher pode ser usado para gerenciar containers Docker em uma variedade de plataformas, incluindo Kubernetes, Docker Swarm e Mesos. Rancher fornece recursos para monitorar containers Docker, como monitoramento de desempenho, monitoramento de integridade e monitoramento de segurança.

Certinho pessoal? Mas ficou interessado em saber sobre os demais itens? Se quiser saber leia a seguir a explicação breve sobre cada um deles: Webhooker é uma ferramenta de integração contínua e contínua (CI/CD) que pode ser usada para automatizar o processo de lançamento de aplicativos. Swagger é uma ferramenta de documentação de APIs que pode ser usada para documentar APIs REST. Zuul é um gateway

de API que pode ser usado para rotear o tráfego para diferentes APIs. Flyway é uma ferramenta de gerenciamento de banco de dados que pode ser usada para gerenciar as migrações de banco de dados.

Gabarito: Letra A

---

19. (FCC – TRT - 22ª Região (PI) – 2022) O Docker tornou muito mais fácil para os desenvolvedores entender e usar a tecnologia de contêineres. Para isso, oferece diversos recursos ou ferramentas, dentre as quais encontram-se:

I. Ferramenta de clustering e scheduling para contêineres do Docker, que permite que os administradores e desenvolvedores de TI possam estabelecer e gerenciar um cluster de nós do Docker como um único sistema virtual.

II. Reúne instruções necessárias para construir uma imagem de contêiner.

III. Aplicativo para plataforma Mac ou Windows que permite criar e compartilhar microsserviços e containerized applications. Inclui diversas ferramentas como o cliente Docker, o Docker Compose, o Docker Content Trust, o Kubernetes e o Credential Helper.

Os itens I, II e III correspondem, correta e respectivamente, a

a) Docherd - Docker Compose - Docker Hub.
b) Docker Engine - Docker Compose - Docker Desktop.
c) Docker Build - Docherd - Docker Swam.
d) Docker Engine - Dockerfile - Docker Hub.
e) Docker Swarm - Dockerfile - Docker Desktop.

Comentários:

A resposta correta é a letra (e).

Item I: Docker Swarm é uma ferramenta de clustering e scheduling para contêineres do Docker. Ele permite que os administradores e desenvolvedores possam estabelecer e gerenciar um cluster de nós do Docker como um único sistema virtual.

Item II: Dockerfile é um arquivo que contém instruções necessárias para construir uma imagem de contêiner. Ele especifica os passos necessários para criar uma imagem de contêiner, incluindo a instalação de software, a configuração de ambientes e a definição de variáveis de ambiente.

Item III: Docker Desktop é uma interface gráfica de usuário para o Docker Engine. Ele fornece uma interface gráfica para gerenciar contêineres Docker, imagens Docker e repositórios Docker.

Gabarito: Letra E

---

20. (CEBRASPE – BNB– 2022) Julgue o seguinte item, relativos a containers de aplicação.

Caso seja necessário informar ao Docker que um container deve escutar na porta de rede 80 do TCP, o comando correto no DockerFile é o seguinte.

EXPOSE 80/tcp

Comentários:

Perfeita questão! A instrução EXPOSE é usada para definir as portas que serão expostas pela imagem.

Gabarito: Correto

---

21. (FCC – TRT - 19ª Região (AL) – 2022) A fim de utilizar o Docker em sua organização, um Analista necessitou conhecer os principais componentes dessa plataforma, tais como:

I. Software que roda na máquina onde o Docker está instalado. Recebe comandos do cliente a partir de Command Line Interfaces ou API's REST.
II. Mecanismo usado para criar imagens e containers.
III. Coleção de imagens hospedadas e rotuladas que juntas permitem a criação do sistema de arquivos de um container. Pode ser público ou privado.
IV. Repositório usado para hospedar e baixar diversas imagens. Pode ser visto como uma plataforma de Software as a Service (SaaS) de compartilhamento e gerenciamento de imagens.

Os itens de I a IV correspondem, correta e respectivamente, a Docker

a) Conteiner – Compose – Swarm – Image.
b) Daemon – Image – Container – Registry.
c) Engine – Swarm – Hub – Registry.
d) Daemon – Engine – Registry – Hub.
e) Conteiner – Engine – Compose – Hub.

Comentários:

Pessoal, vamos analisar cada item: Item I: Daemon é o software que roda na máquina onde o Docker está instalado. Ele recebe comandos do cliente a partir de Command Line Interfaces ou API's REST.
Item II: Engine é o mecanismo usado para criar imagens e containers. Ele é responsável por gerenciar os recursos do sistema operacional, como memória, CPU e armazenamento, e por executar os containers.
Item III: Registry é uma coleção de imagens hospedadas e rotuladas que juntas permitem a criação do sistema de arquivos de um container. Pode ser público ou privado.
Item IV: Hub é um repositório usado para hospedar e baixar diversas imagens. Pode ser visto como uma plataforma de Software as a Service (SaaS) de compartilhamento e gerenciamento de imagens.

Gabarito: Letra D

---

22. (FCC – TJ-CE – 2022) No Linux, em condições ideais, o Docker é disponibilizado com três redes por padrão. Essas redes oferecem configurações específicas para o gerenciamento do tráfego de dados e

a) a rede None tem como objetivo entregar para o container todas as interfaces existentes no docker none. Isso agiliza a entrega dos pacotes, uma vez que não há host no caminho das mensagens. Mas o uso de um host pode ser importante para a segurança e a gerência do tráfego.
b) Bridge é a rede padrão para qualquer container iniciado no Docker, a menos que seja, explicitamente, associada outra rede a ele. Os containers da rede default bridge podem acessar uns aos outros somente através de seus endereços IP, a menos que se utilize a opção --link
c) deve-se, para visualizar as redes no Docker, utilizar o comando: docker network -show
d) todos os containers que estão na rede None poderão se comunicar via protocolo TCP/IP. Se uma pessoa souber qual é o endereço IP do container que deseja conectar, é possível enviar tráfego para ele, pois estão todos na mesma rede IP (172.17.0.0/24).
e) a rede Host tem como objetivo isolar o container para comunicações externas. A rede não recebe qualquer interface para comunicação externa. A única interface de rede IP será a localhost. Essa rede, normalmente, é utilizada para containers que manipulam apenas arquivos de backup, sem necessidade de enviá-los via rede para outro local.

Comentários:

Pessoal, a resposta correta é a letra B. As redes padrão do Docker no Linux são Bridge, Host e None. A rede Bridge é a rede padrão para qualquer container iniciado no Docker, a menos que seja, explicitamente, associada outra rede a ele. Os containers da rede default bridge podem acessar uns aos outros somente através de seus endereços IP, a menos que se utilize a opção --link. Por outro lado, a rede Host é uma rede que permite que um container compartilhe a rede do host. Os containers da rede host podem acessar o host diretamente e outros containers na rede host. Por fim, a rede None: É uma rede que não permite que um container se conecte a nenhuma outra rede. Os containers da rede none não podem se comunicar com outros containers ou com o host. Portanto, nosso gabarito é a Letra B.

Gabarito: Letra B

---

23. **(CEBRASPE – DPE-RO – 2022)** A orquestração automatiza a implantação, o gerenciamento, a escala e a rede dos contêineres. As ferramentas de orquestração de contêineres fornecem um framework para gerenciar arquiteturas de microsserviços e contêineres em escala, e muitas delas são usadas no gerenciamento do ciclo de vida dos contêineres; entre elas, o Docker Swarm é uma plataforma

a) que permite utilizar diversos recursos e ferramentas, como Apache e PHP, porém tudo rodando em um mesmo sistema operacional.
b) de código aberto criada pelo Google para operações de implantação de contêiner, aumento e redução e automação em clusters de hosts.
c) de orquestração de contêiner de código aberto, sendo o mecanismo de clusterização nativo para e pelo Docker, utilizando sua mesma linha de comando.
d) que roda sobre o Kubernetes instalado em sistema operacional na versão Enterprise da Red Hat, agregando opções de monitoramento, integração e entrega contínua.
e) usada pela Amazon para fornecer outros serviços aos clientes, como DNS, balanceamento, segurança e monitoramento, se integrando nativamente.

Comentários:

A orquestração automatizada é uma parte fundamental na gestão de contêineres e arquiteturas de microsserviços, facilitando a implantação, gerenciamento, escalabilidade e rede dos contêineres. Ferramentas de orquestração de contêineres oferecem um framework que permite administrar ambientes complexos em larga escala. Dentre essas ferramentas, o Docker Swarm é destacado como uma plataforma de código aberto voltada para a orquestração de contêineres.

O Docker Swarm proporciona uma abordagem nativa de orquestração para os contêineres Docker. Essa plataforma transforma várias instâncias do Docker em um host virtual coeso, formando um cluster do Docker Swarm. Esse cluster é constituído por diferentes elementos, incluindo os nós que executam os contêineres, os serviços que definem como os contêineres devem ser implantados e escalados, bem como os balanceadores de carga que garantem a distribuição adequada do tráfego.

No entanto, é relevante mencionar que o Docker Swarm tem seu uso específico e suas próprias vantagens e desvantagens em comparação com outras soluções, como o Kubernetes. Ambas as plataformas são populares para a orquestração de contêineres, cada uma com suas características e casos de uso distintos. Portanto, a resposta correta é a opção c) de orquestração de contêiner de código aberto, sendo o mecanismo de clusterização nativo para e pelo Docker, utilizando sua mesma linha de comando.

Gabarito: Letra C

---

24. (Quadrix – CRA-PR – 2022) No Docker, a comunicação entre containers somente é possível se forem criados dois Net Namespaces iguais.

Comentários:

A afirmação é incorreta. O Net Namespace permite que cada container possua sua interface de rede e portas. Para que seja possível a comunicação entre os containers, é necessário criar dois Net Namespaces diferentes, um responsável pela interface do container (normalmente utilizamos o mesmo nome das interfaces convencionais do Linux, por exemplo, a etho) e outro responsável por uma interface do host, normalmente chamada de veth* (veth + um identificador aleatório). Essas duas interfaces estão linkadas através da bridge Dockero no host, que permite a comunicação entre os containers através de roteamento de pacotes.

Gabarito: Errado

---

25. (Quadrix – CRA-PR – 2022) Para se instalar o Docker em um ambiente Linux, é necessário que o processador seja de 32 bits e que a versão do kernel seja inferior a 3.6.

Comentários:

A afirmação é errada. O Docker é compatível com processadores de 64 bits e kernels do Linux a partir da versão 3.10. A versão mais recente do Docker é a 20.10.8, lançada em 17 de junho de 2023. A documentação do Docker especifica que esta versão é compatível com os seguintes sistemas operacionais Linux: kernel 3.10 ou superior, Windows: Windows Server 2016 ou superior e macOS: macOS 10.13 ou superior.

Gabarito: Errado

---

26. (Quadrix – CRA-PR – 2022) Um dos namespaces utilizados pelo Docker é o PID namespace, o qual permite que cada container tenha seus próprios identificadores de processos.

Comentários:

A afirmação é correta. O PID namespace é um dos namespaces utilizados pelo Docker. Ele permite que cada container tenha seu próprio espaço de identificação de processos. O espaço de identificação de processos é um conjunto de identificadores exclusivos que são usados para identificar processos no sistema operacional. O PID namespace permite que cada container tenha seu próprio espaço de identificação de processos, o que significa que os processos de um container não podem ser vistos ou acessados por processos de outros containers. O PID namespace é uma ferramenta importante para a segurança e a confiabilidade dos containers. Ele ajuda a impedir que os processos de um container sejam afetados por problemas em outros containers.

Gabarito: Correto

---

27. (CEFET-MG – CEFET-MG – 2021) A partir do acesso ao terminal de um servidor Debian GNU/Linux com o Docker Engine e Docker Compose devidamente instalados, considere:

· O usuário logado no terminal é o root
· A saída do comando pwd é: /home/user/projeto
· A saída do comando docker image ls é:

root@server:/home/user/projeto# docker
image ls
REPOSITORY TAG IMAGE ID CREATED SIZE

A saída do comando ls -lha é:

root@server:/home/user/projeto# ls -lha
drwxrwxrwx 1 user user 512 Aug 9 10:51 .
drwxrwxrwx 1 user user 512 Aug 9 10:49 ..
-rwxrwxrwx 1 user user 190 Aug 9 10:51 Dockerfile
-rwxrwxrwx 1 user user 121 Aug 9 10:51 docker-compose.yml

O conteúdo do arquivo Dockerfile é:

FROM debian:stable
RUN apt-get update && apt-get install -y apache2
EXPOSE 80
VOLUME [“/var/www” , “/var/log/apache2” , “/etc/apache2”]
ENTRYPOINT [“/usr/sbin/apache2ctl” , “-D” , “FOREGROUND”]

O conteúdo do arquivo docker-compose.yml é

version: "3.9"

```
services:
 web:
  build: .
  ports:
   - "8080:80"
```

Para se colocar em execução, em segundo plano, um contêiner baseado na imagem especificada pelo arquivo Dockerfile, publicando a porta 8080 do host para a porta 80 do contêiner, é/são suficiente(s) o(s) comando(s):

a) docker build -t imagem-prova:latest . && docker container run -it -p 8080:80 imagem-prova:latest
b) docker build -t imagem-prova:latest . && docker container run -d -p 80:8080 imagem-prova:latest
c) docker container run -d -p 8080:80 imagem-prova:latest
d) docker-compose -f docker-compose.yml up
e) docker-compose up -d

Comentários:

O Docker Compose é uma ferramenta para definir e executar aplicativos Docker com base em arquivos de composição, como o docker-compose.yml. O arquivo docker-compose.yml já contém as configurações necessárias, incluindo a definição do serviço "web" que usa o Dockerfile no diretório atual para construir a imagem. Ao executar docker-compose up -d, você inicia o serviço "web" em segundo plano, seguindo as configurações definidas no arquivo de composição. Isso inclui o mapeamento da porta 8080 do host para a porta 80 do contêiner. Portanto, a opção E é nosso gabarito e irá para colocar em execução o contêiner com base nas configurações fornecidas nos arquivos de composição. Isso simplifica o processo, pois o Docker Compose lida com a construção da imagem e a execução do contêiner com as configurações desejadas. Além disso, as alternativas (A), (B) e (C) não são suficientes para iniciar o contêiner. A alternativa (A) irá construir a imagem imagem-prova:latest e, em seguida, iniciar um contêiner baseado nesta imagem. No entanto, o contêiner não será iniciado em segundo plano. A alternativa (B) irá construir a imagem imagem-prova:latest e, em seguida, iniciar um contêiner baseado nesta imagem, publicando a porta 80 do host para a porta 8080 do contêiner. No entanto, o contêiner não será iniciado em segundo plano. A alternativa (C) irá iniciar um contêiner baseado na imagem imagem-prova:latest, publicando a porta 8080 do host para a porta 80 do contêiner. No entanto, a imagem imagem-prova:latest não existe.

Gabarito: Letra E

---

28. (CEBRASPE – SERPRO – 2021). Considere o seguinte conteúdo de um dockerfile.

```
FROM rhel7:latest
USER root

MAINTAINER Joao

RUN yum -y install deltarpm yum-utils --
disablerepo=*-eus-* --disablerepo=*-htb-* *-
sjis-*\
```

```
    --disablerepo=*-ha-* --disablerepo=*-rt-* --
disablerepo=*-lb-* --disablerepo=*-rs-* --
disablerepo=*-sap-*


RUN yum-config-manager --disable *-eus-* *-htb-*
*-ha-* *-rt-* *-lb-* *-rs-* *-sap-* *-sjis* >
/dev/null


RUN yum install httpd procps-ng MySQL-python -y


ADD action /var/www/cgi-bin/action
RUN echo "PassEnv DB_SERVICE_SERVICE_HOST" >>
/etc/httpd/conf/httpd.conf
RUN chown root:apache /var/www/cgi-bin/action
RUN chmod 755 /var/www/cgi-bin/action
RUN echo "Pagina Inicial" >
/var/www/html/index.html
EXPOSE 80

CMD mkdir /run/httpd ; /usr/sbin/httpd -D
FOREGROUND
```

Tendo como referência essas informações, julgue o item a seguir.

A imagem base do container é um Red Hat Linux.

Comentários:

Sim, o item está correto. A linha FROM rhel7:latest no Dockerfile especifica que a imagem base do container é um Red Hat Enterprise Linux 7 (RHEL 7).

O Dockerfile é um arquivo de texto que descreve como criar uma imagem Docker. Ele é dividido em instruções, cada uma das quais executa uma ação específica. As instruções no Dockerfile acima fazem o seguinte:

FROM rhel7:latest: Define a imagem base do container como um RHEL 7.
USER root: Define o usuário que será usado para executar os comandos subsequentes.
MAINTAINER Joao: Especifica o nome do autor do Dockerfile.
RUN yum -y install deltarpm yum-utils --disablerepo=*-eus-* --disablerepo=*-htb-* *-sjis-* --disablerepo=*-ha-* --disablerepo=*-rt-* --disablerepo=*-lb-* --disablerepo=*-rs-* --disablerepo=*-sap-*: Instala os pacotes deltarpm e yum-utils e desabilita os repositórios de software que não são necessários.
RUN yum-config-manager --disable *-eus-* *-htb-* *-ha-* *-rt-* *-lb-* *-rs-* *-sap-* *-sjis* > /dev/null: Grava uma mensagem de confirmação no arquivo /dev/null.
RUN yum install httpd procps-ng MySQL-python -y: Instala os pacotes httpd, procps-ng e MySQL-python.

ADD action /var/www/cgi-bin/action: Copia o arquivo action para o diretório /var/www/cgi-bin.
RUN echo "PassEnv DB_SERVICE_SERVICE_HOST" >> /etc/httpd/conf/httpd.conf: Adiciona uma linha ao arquivo /etc/httpd/conf/httpd.conf que define a variável de ambiente DB_SERVICE_SERVICE_HOST.
RUN chown root:apache /var/www/cgi-bin/action: Altera a propriedade do arquivo /var/www/cgi-bin/action para o usuário root e o grupo apache.
RUN chmod 755 /var/www/cgi-bin/action: Altera as permissões do arquivo /var/www/cgi-bin/action para 755.
RUN echo "Pagina Inicial" > /var/www/html/index.html: Cria um arquivo chamado index.html no diretório /var/www/html e o preenche com o texto "Pagina Inicial".
EXPOSE 80: Expor a porta 80 do container para o host.
CMD mkdir /run/httpd ; /usr/sbin/httpd -D FOREGROUND: Define o comando que será executado quando o container for iniciado.
Portanto, o item está correto. A imagem base do container é um Red Hat Enterprise Linux 7 (RHEL 7).

Gabarito: Correto

---

29. (UFMT – UFMT – 2021) Sobre Contêiner no Sistema Operacional Linux Ubuntu Server 18.04, analise o comando a seguir.

docker run --name web -p 80:80 -p 443:443 -p 65533:22 -ti ubuntu bash

Marque V para as afirmativas verdadeiras e F para as falsas.
( ) A opção --name indica o nome que o contêiner terá, nesse caso, web.
( ) Ao executar esse comando, o docker criará um contêiner baseado em uma imagem do ubuntu.
( ) A opção -p indica que o contêiner deve debugar essas range de portas apresentadas na frente do parâmetro.
Assinale a sequência correta.

a) F, F, V
b) V, F, F
c) V, V, F
d) F, V, V

Comentários:

A sequência correta é (c). As afirmativas (I - A opção --name indica o nome que o contêiner terá, nesse caso, web.) e (II - Ao executar esse comando, o docker criará um contêiner baseado em uma imagem do ubuntu.) são verdadeiras, enquanto a afirmativa (III - A opção -p indica que o contêiner deve debugar essas range de portas apresentadas na frente do parâmetro) é falsa.

A opção -p é usada para mapear portas do host para portas do contêiner. No comando fornecido, as portas 80, 443 e 22 do host serão mapeadas para as portas 80, 443 e 65533 do contêiner, respectivamente. Isso significa que o contêiner poderá ser acessado pelo host a partir dessas portas. A sequência correta é V,V.

Gabarito: Letra C

---

30. (TJ-RN – TJ-RN – 2020) Os volumes são mecanismos utilizados para persistir os dados gerados e usados pelos containers do Docker. Embora as montagens de ligação dependam da estrutura de diretórios da máquina host, os volumes são completamente gerenciados pelo Docker.

Considerando que um analista queira criar um volume de nome my-volume dentro de um docker, ele deve executar o comando

a) docker volume create my-volume
b) docker create volume my-volume
c) docker run create volume my-volume
d) docker create run volume my-volume

Comentários:

A resposta correta é (a). O comando docker volume create é usado para criar um volume Docker. O parâmetro my-volume especifica o nome do volume que será criado.

As outras alternativas são incorretas porque:
(b): O comando docker create é usado para criar um contêiner Docker.
(c): O comando docker run create não existe.
(d): O comando docker create run volume my-volume é uma combinação dos comandos docker create e docker volume create, que não é necessária.
Portanto, a alternativa (a) é a única correta.

Gabarito: Letra A

LISTA DE QUESTÕES

1. **(CEBRASPE – MPE-RO – 2023)** Assinale a opção correspondente ao comando, em Docker, que permite ao usuário fazer o download de uma imagem específica ou um conjunto de imagens do Docker Hub.

   a) docker attach
   b) docker node ls
   c) docker pull
   d) docker rmi
   e) docker rm / redis

2. **(CEBRASPE – MPE-RO – 2023)** Assinale a opção em que é apresentado o comando que permite listar todos os nós que o gerenciador do Docker Swarm conhece.

   a) docker attach
   b) docker node ls
   c) docker pull
   d) docker rmi
   e) docker rm /redis

3. **(CEBRASPE – SEFIN de Fortaleza - CE – 2023)** Tendo em vista que, no atual cenário de desenvolvimento de aplicações web, é essencial considerar princípios, como consistência e escalabilidade, e práticas, como automação do processo de implantação e integração do código-fonte, julgue o item subsequente.

   Um exemplo prático de conteinerização de aplicação é a utilização do Docker para criar um ambiente consistente; nesse caso, é correto criar um arquivo Dockerfile por meio do comando docker build.

4. **(FGV – TJ-RN – 2023)** A analista Maria gerencia a aplicação WebJus. A WebJus requer a execução de dois containers Docker que necessitam de comunicação entre si. No entanto, para fins de balanceamento de carga, cada container da WebJus foi alocado em um servidor diferente. A fim de configurar a comunicação entre os containers da WebJus de forma simples, Maria criou a rede Docker NetJus, que é capaz de conectar containers rodando em servidores diferentes.

   A NetJus utiliza o driver de rede Docker:

   a) host;
   b) ipvlan;
   c) bridge;
   d) overlay;
   e) macvlan.

5. **(UFPR – IF-PR – 2023)** O comando docker run cria e executa um novo contêiner a partir de uma imagem. Para a criação desse contêiner em segundo plano, utiliza-se o comando:

   a) docker container run -b

b) docker container run -p
c) docker container run -a
d) docker container run -d
e) docker container run -s

6. **(INSTITUTO AOCP – IF-MA – 2023)** Osnei é Técnico em Tecnologia da Informação e está trabalhando em um sistema para a instituição em que atua. Ele está utilizando a aplicação de Docker para seu desenvolvimento. Qual comando pode ser utilizado para exibir as versões de API, Client e Server do hots?

a) Docker version.
b) Docker run.
c) Docker api.
d) Docker cliente.
e) Docker server.

7. **(UFPR – IF-PR – 2023)** O comando docker run cria e executa um novo contêiner a partir de uma imagem. Para a criação desse contêiner em segundo plano, utiliza-se o comando:

a) docker container run -b
b) docker container run -p
c) docker container run -a
d) docker container run -d
e) docker container run -s

8. **(FGV – SEFAZ-AM – 2022)** Leia o fragmento a seguir.
"A plataforma Docker usa uma arquitetura do tipo _____. O cliente Docker conversa com o daemon do Docker, que constrói, executa e distribui _____ Docker. O cliente e o daemon do Docker podem ser executados em um mesmo sistema ou se conectar um cliente do Docker a um daemon remoto. O cliente Docker e o daemon se comunicam usando _____ ou uma interface de redes."

Assinale a opção cujos itens completam corretamente as lacunas do fragmento acima, na ordem apresentada.

a) MVC – imagens – chamadas RPC ou bluetooth.
b) thin client – contêineres – wireless ou bluetooth.
c) serverless – componentes – chamadas MPI ou RPC.
d) cliente-servidor – contêineres – API REST ou soquetes UNIX.
e) mesh app and service – imagens – API RESTFULL ou wireless.

9. **(FGV – TJ-DFT – 2022)** A analista Sara modificou alguns arquivos do container Docker TJSiteContainer que se encontra em execução. Para criar uma imagem Docker a partir do estado atual de TJSiteContainer, a fim de persistirem as modificações efetuadas em seus arquivos, Sara deve utilizar o comando:

a) docker create
b) docker diff

c) docker export
d) docker image save
e) docker commit

10. (FGV – SEFAZ-AM – 2022) Um Docker Hub consiste em um índice e registro do Docker. Com relação ao registro do Docker, analise as afirmativas a seguir e assinale (V) para a verdadeira e (F) para a falsa.

( ) Possui recursos avançados que incluem bugsnag, new relic e cors.

( ) Suporta diferentes tipos de back-end de armazenamento de arquivos em nuvem ou sistema de arquivos local.

( ) Armazena dados referentes às contas de usuários em banco de dados local.

As afirmativas são, na ordem apresentada, respectivamente

a) F - V - V.
b) V - V - F.
c) F - F - V.
d) F - V - F.
e) V - F - F.

11. (FGV – CGU – 2022) Uma das estratégias para reduzir o tamanho de imagens Docker consiste em:

a) combinar comandos RUN em um único comando;
b) substituir a imagem base por uma versão mais recente;
c) separar comandos RUN complexos em comandos menores;
d) reordenar os comandos de forma que o cache seja utilizado com maior frequência;
e) compactar arquivos a serem copiados para a imagem e descompactá-los durante a sua geração

12. (FGV – MPE-GO – 2022) Uma equipe de especialistas está implantando uma nova aplicação web que opera em conjunto com uma base de dados. A equipe resolveu utilizar o Docker para esta implantação isolando a aplicação em um container e a base de dados em outro container.

Para definir os dois containers e configurar sua comunicação através de um único arquivo, a equipe deve usar a ferramenta Docker

a) File.
b) Scan.
c) Buildx.
d) Compose.
e) Hub.

13.) A plataforma Docker pode criar imagens automaticamente, executando as instruções de um arquivo Dockerfile.

A primeira instrução presente em um Dockerfile é denominada

a) ADD.
b) RUN.
c) FROM.
d) EXPOSE.
e) WORKDIR.

14. **(FGV – TRT - 16ª REGIÃO (MA) – 2022)** Docker é uma plataforma que permite criar e compartilhar aplicativos e microsserviços em contêineres.

O comando utilizado para executar um contêiner novo é o

a) docker composes.
b) docker create.
c) docker build.
d) docker exec.
e) docker run.

15. **(FGV – MPE-GO – 2022)** Um assistente programador está avaliando preventivamente o comportamento de um servidor do MP-GO que executa containers Docker. O assistente deseja verificar os nomes de todos os containers que estão parados e de todos os containers que estão em execução em uma única saída no terminal do servidor.

Para realizar essa verificação usando um simples comando no terminal, o assistente deve usar o comando

a) docker ps -a
b) docker ps -l
c) docker ps -s
d) docker ps -q
e) docker ps

16. **(FGV – SEFAZ-AM – 2022)** Com relação à segurança dos contêineres Docker, analise as afirmativas a seguir e assinale (V) para a verdadeira e (F) para a falsa.

( ) Possuem isolamento no nível do processo no sistema operacional e isolamento adicional, usando recursos especiais tais como namespaces e cgroups.

( ) Aproveitam-se dos mecanismos interprocess communication padrão, tais como sinais, pipes e sockets, onde cada contêiner possui a sua própria network stack.

( ) Os contêineres possuem um multi level security no sistema operacional do host e os recursos físicos que são gerenciados por um hypervisor compartilhado.

As afirmativas são, na ordem apresentada, respectivamente,

a) F - V - V.

b) F - F - V.
c) V - F - V.
d) V - F - F.
e) V - V - F.

17. **(FGV – TRT - 18ª Região (GO) – 2022)** Um técnico deseja usar o Keycloak no Docker, instalado e em condições ideais.
I 8080:8080 -e KEYCLOAK_ADMIN=admin -e KEYCLOAK_ADMIN_PASSWORD=admin quay.io/keycloak/keycloak:20.0.0 start-dev

Para iniciar o Keycloak exposto na porta local 8080, criando um usuário inicial admin, com senha admin, a lacuna I deve ser preenchida por:

a) docker run -v
b) docker container start on port
c) docker run -p
d) docker start --port
e) docker container run -d port

18. **(FCC – TRT - 23ª REGIÃO (MT) – 2022)** Um Técnico precisa administrar e monitorar containers Docker em produção. Para isso pode utilizar a ferramenta

a) Rancher
b) Webhooker
c) Swagger
d) Zuul
e) Flyway

19. **(FCC – TRT - 22ª Região (PI) – 2022)** O Docker tornou muito mais fácil para os desenvolvedores entender e usar a tecnologia de contêineres. Para isso, oferece diversos recursos ou ferramentas, dentre as quais encontram-se:

I. Ferramenta de clustering e scheduling para contêineres do Docker, que permite que os administradores e desenvolvedores de TI possam estabelecer e gerenciar um cluster de nós do Docker como um único sistema virtual.

II. Reúne instruções necessárias para construir uma imagem de contêiner.

III. Aplicativo para plataforma Mac ou Windows que permite criar e compartilhar microsserviços e containerized applications. Inclui diversas ferramentas como o cliente Docker, o Docker Compose, o Docker Content Trust, o Kubernetes e o Credential Helper.

Os itens I, II e III correspondem, correta e respectivamente, a

a) Docherd - Docker Compose - Docker Hub.
b) Docker Engine - Docker Compose - Docker Desktop.

c) Docker Build - Docherd - Docker Swam.
d) Docker Engine - Dockerfile - Docker Hub.
e) Docker Swarm - Dockerfile - Docker Desktop.

20. **(CEBRASPE – BNB– 2022)** Julgue o seguinte item, relativos a containers de aplicação.

Caso seja necessário informar ao Docker que um container deve escutar na porta de rede 80 do TCP, o comando correto no DockerFile é o seguinte.

EXPOSE 80/tcp

21. **(FCC – TRT - 19ª Região (AL) – 2022)** A fim de utilizar o Docker em sua organização, um Analista necessitou conhecer os principais componentes dessa plataforma, tais como:

I. Software que roda na máquina onde o Docker está instalado. Recebe comandos do cliente a partir de Command Line Interfaces ou API's REST.
II. Mecanismo usado para criar imagens e containers.
III. Coleção de imagens hospedadas e rotuladas que juntas permitem a criação do sistema de arquivos de um container. Pode ser público ou privado.
IV. Repositório usado para hospedar e baixar diversas imagens. Pode ser visto como uma plataforma de Software as a Service (SaaS) de compartilhamento e gerenciamento de imagens.

Os itens de I a IV correspondem, correta e respectivamente, a Docker

a) Conteiner – Compose – Swarm – Image.
b) Daemon – Image – Container – Registry.
c) Engine – Swarm – Hub – Registry.
d) Daemon – Engine – Registry – Hub.
e) Conteiner – Engine – Compose – Hub.

22. **(FCC – TJ-CE – 2022)** No Linux, em condições ideais, o Docker é disponibilizado com três redes por padrão. Essas redes oferecem configurações específicas para o gerenciamento do tráfego de dados e

a) a rede None tem como objetivo entregar para o container todas as interfaces existentes no docker none. Isso agiliza a entrega dos pacotes, uma vez que não há host no caminho das mensagens. Mas o uso de um host pode ser importante para a segurança e a gerência do tráfego.
b) Bridge é a rede padrão para qualquer container iniciado no Docker, a menos que seja, explicitamente, associada outra rede a ele. Os containers da rede default bridge podem acessar uns aos outros somente através de seus endereços IP, a menos que se utilize a opção --link
c) deve-se, para visualizar as redes no Docker, utilizar o comando: docker network -show
d) todos os containers que estão na rede None poderão se comunicar via protocolo TCP/IP. Se uma pessoa souber qual é o endereço IP do container que deseja conectar, é possível enviar tráfego para ele, pois estão todos na mesma rede IP (172.17.0.0/24).
e) a rede Host tem como objetivo isolar o container para comunicações externas. A rede não recebe qualquer interface para comunicação externa. A única interface de rede IP será a localhost. Essa rede, normalmente, é utilizada para containers que manipulam apenas arquivos de backup, sem necessidade de enviá-los via rede para outro local.

23. **(CEBRASPE – DPE-RO – 2022)** A orquestração automatiza a implantação, o gerenciamento, a escala e a rede dos contêineres. As ferramentas de orquestração de contêineres fornecem um framework para gerenciar arquiteturas de microsserviços e contêineres em escala, e muitas delas são usadas no gerenciamento do ciclo de vida dos contêineres; entre elas, o Docker Swarm é uma plataforma

a) que permite utilizar diversos recursos e ferramentas, como Apache e PHP, porém tudo rodando em um mesmo sistema operacional.
b) de código aberto criada pelo Google para operações de implantação de contêiner, aumento e redução e automação em clusters de hosts.
c) de orquestração de contêiner de código aberto, sendo o mecanismo de clusterização nativo para e pelo Docker, utilizando sua mesma linha de comando.
d) que roda sobre o Kubernetes instalado em sistema operacional na versão Enterprise da Red Hat, agregando opções de monitoramento, integração e entrega contínua.
e) usada pela Amazon para fornecer outros serviços aos clientes, como DNS, balanceamento, segurança e monitoramento, se integrando nativamente.

24. **(Quadrix – CRA-PR – 2022)** No Docker, a comunicação entre containers somente é possível se forem criados dois Net Namespaces iguais.

25. **(Quadrix – CRA-PR – 2022)** Para se instalar o Docker em um ambiente Linux, é necessário que o processador seja de 32 bits e que a versão do kernel seja inferior a 3.6.

26. **(Quadrix – CRA-PR – 2022)** Um dos namespaces utilizados pelo Docker é o PID namespace, o qual permite que cada container tenha seus próprios identificadores de processos.

27. **(CEFET-MG – CEFET-MG – 2021)** A partir do acesso ao terminal de um servidor Debian GNU/Linux com o Docker Engine e Docker Compose devidamente instalados, considere:

· O usuário logado no terminal é o root
· A saída do comando pwd é: /home/user/projeto
· A saída do comando docker image ls é:

root@server:/home/user/projeto# docker
image ls
REPOSITORY TAG IMAGE ID CREATED SIZE

A saída do comando ls -lha é:

root@server:/home/user/projeto# ls -lha
drwxrwxrwx 1 user user 512 Aug 9 10:51 .
drwxrwxrwx 1 user user 512 Aug 9 10:49 ..
-rwxrwxrwx 1 user user 190 Aug 9 10:51 Dockerfile
-rwxrwxrwx 1 user user 121 Aug 9 10:51 docker-compose.yml

O conteúdo do arquivo Dockerfile é:

```
FROM debian:stable
RUN apt-get update && apt-get install -y apache2
EXPOSE 80
VOLUME [ “/var/www”, “/var/log/apache2”, “/etc/apache2” ]
ENTRYPOINT [ “/usr/sbin/apache2ctl”, “-D”, “FOREGROUND” ]
```

O conteúdo do arquivo docker-compose.yml é

```
version: "3.9"
services:
 web:
   build: .
   ports:
    - "8080:80"
```

Para se colocar em execução, em segundo plano, um contêiner baseado na imagem especificada pelo arquivo Dockerfile, publicando a porta 8080 do host para a porta 80 do contêiner, é/são suficiente(s) o(s) comando(s):

a) docker build -t imagem-prova:latest . && docker container run -it -p 8080:80 imagem-prova:latest
b) docker build -t imagem-prova:latest . && docker container run -d -p 80:8080 imagem-prova:latest
c) docker container run -d -p 8080:80 imagem-prova:latest
d) docker-compose -f docker-compose.yml up
e) docker-compose up -d

28. (CEBRASPE – SERPRO – 2021). Considere o seguinte conteúdo de um dockerfile.

```
FROM rhel7:latest
USER root

MAINTAINER Joao

RUN yum -y install deltarpm yum-utils --
disablerepo=*-eus-* --disablerepo=*-htb-* *-
sjis-*\

   --disablerepo=*-ha-* --disablerepo=*-rt-* --
disablerepo=*-lb-* --disablerepo=*-rs-* --
disablerepo=*-sap-*


RUN yum-config-manager --disable *-eus-* *-htb-*
*-ha-* *-rt-* *-lb-* *-rs-* *-sap-* *-sjis* >
/dev/null


RUN yum install httpd procps-ng MySQL-python -y
```

```
ADD action /var/www/cgi-bin/action
RUN echo "PassEnv DB_SERVICE_SERVICE_HOST" >>
/etc/httpd/conf/httpd.conf
RUN chown root:apache /var/www/cgi-bin/action
RUN chmod 755 /var/www/cgi-bin/action
RUN echo "Pagina Inicial" >
/var/www/html/index.html
EXPOSE 80

CMD mkdir /run/httpd ; /usr/sbin/httpd -D
FOREGROUND
```

Tendo como referência essas informações, julgue o item a seguir.

A imagem base do container é um Red Hat Linux.

29. **(UFMT – UFMT – 2021)** Sobre Contêiner no Sistema Operacional Linux Ubuntu Server 18.04, analise o comando a seguir.

docker run --name web -p 80:80 -p 443:443 -p 65533:22 -ti ubuntu bash

Marque V para as afirmativas verdadeiras e F para as falsas.
( ) A opção --name indica o nome que o contêiner terá, nesse caso, web.
( ) Ao executar esse comando, o docker criará um contêiner baseado em uma imagem do ubuntu.
( ) A opção -p indica que o contêiner deve debugar essas range de portas apresentadas na frente do parâmetro.
Assinale a sequência correta.

a) F, F, V
b) V, F, F
c) V, V, F
d) F, V, V

30. **(TJ-RN – TJ-RN – 2020)** Os volumes são mecanismos utilizados para persistir os dados gerados e usados pelos containers do Docker. Embora as montagens de ligação dependam da estrutura de diretórios da máquina host, os volumes são completamente gerenciados pelo Docker.

Considerando que um analista queira criar um volume de nome my-volume dentro de um docker, ele deve executar o comando

a) docker volume create my-volume
b) docker create volume my-volume
c) docker run create volume my-volume
d) docker create run volume my-volume

GABARITO

1. Letra C
2. Letra B
3. Errado
4. Letra D
5. Letra D
6. Letra A
7. Letra D
8. Letra D
9. Letra E
10. Letra B
11. Letra A
12. Letra D
13. Letra C
14. Letra E
15. Letra A
16. Letra E
17. Letra C
18. Letra A
19. Letra E
20. Correto
21. Letra D
22. Letra B
23. Letra C
24. Errado
25. Errado
26. Correto
27. Letra E
28. Correto
29. Letra C
30. Letra A

Sumário

Apresentação da Aula 2
Kubernetes 4
Conceitos Básicos 4
Por que você precisa do Kubernetes e o que ele pode fazer 5
Os principais benefícios do Kubernetes 6
Orquestração de contêineres prontos para produção 9
Objetos no Kubernetes 10
Compreendendo os objetos no Kubernetes 10
API do Kubernetes 12
Grupos de API e Versionamento 12
Arquitetura do Cluster 13
Controladores 17
Contexto 19
Containers 20
Imagens 20
Ambiente de contêineres 24
Hooks de ciclo de vida para os Contêineres 24
Componentes Kubernetes 28
Componentes do Plano de Controle 28
kube-apiserver 29
etcd 29
kube-scheduler 29
kube-controller-manager 31
cloud-controller-manager 31
Componentes do nó 32
kubelet 32
kube-proxy 33
kube-apiserver 33
kube-controller-manager 33
Pod 35
Classes de Qualidade de Serviço de Pods 35
Segurança 37
Controlando o acesso à API do Kubernetes 37
Padrões de Segurança para Pods 38
Utilizando o kubectl exec para obter um shell em um contêiner em execução 40
Plataforma como Serviço e Container como Serviço 46
FaaS 46
OpenFaaS 49
CaaS 50
Referências 53

APRESENTAÇÃO DA AULA

Olá, pessoal! Hoje vamos explorar o mundo Kubernetes e entender como essa tecnologia está revolucionando a maneira como implantamos e gerenciamos aplicativos em ambientes modernos de computação em nuvem. O Kubernetes, frequentemente abreviado como K8s, é uma plataforma de orquestração de contêineres de código aberto que simplifica e automatiza o gerenciamento de aplicações containerizadas.

Imagine um ambiente onde os aplicativos são divididos em unidades autônomas chamadas contêineres, que empacotam o código, as bibliotecas e as dependências necessárias para executar de maneira consistente em qualquer ambiente. Agora, o Kubernetes entra em cena para coordenar e gerenciar esses contêineres de forma eficiente.

Uma das características marcantes do Kubernetes é sua capacidade de automatizar tarefas complexas, como escalonamento automático, distribuição de carga, atualizações de aplicativos e recuperação de falhas. Isso significa que os desenvolvedores podem se concentrar no desenvolvimento de código, enquanto o Kubernetes cuida dos aspectos de implantação, dimensionamento e disponibilidade.

Ao utilizar o Kubernetes, os aplicativos podem ser facilmente dimensionados horizontalmente para lidar com picos de tráfego ou demanda, e podem ser implantados consistentemente em uma variedade de ambientes, desde ambientes locais até várias nuvens públicas. A arquitetura do Kubernetes é baseada em conceitos como nós, pods, serviços e controladores, que trabalham em conjunto para fornecer um ambiente altamente resiliente e flexível.

Além disso, o Kubernetes oferece uma ampla gama de recursos para monitorar e manter a saúde dos aplicativos, garantindo que eles estejam sempre disponíveis e funcionando conforme o esperado. Ferramentas de monitoramento, como Prometheus e Grafana, podem ser integradas para fornecer insights detalhados sobre o desempenho e a utilização dos recursos.

Pelas minhas pesquisas, observei que as bancas buscam muitos conceitos direto da documentação, e muitas vezes COPIAM a exata definição do conceito para as questões (prova). Assim, sugiro fortemente que quando tiver aquele tempinho: antes de dormir, em uma fila, ou

qualquer tempo possível, acesse a documentação do kubernetes e veja alguns conceitos neste link: https://kubernetes.io/docs/concepts/overview/ 😀

KUBERNETES

Conceitos Básicos

Kubernetes (K8s) fornece um ambiente de orquestração eficiente para aplicativos em contêineres, tornando mais fácil o gerenciamento de aplicações complexas. É um produto Open Source utilizado para automatizar a implantação, o dimensionamento e o gerenciamento de aplicativos em contêiner. Ele agrupa contêineres que compõem uma aplicação em unidades lógicas para facilitar o gerenciamento e a descoberta de serviço.

kubernetes

O Kubernetes é fundamentado em uma história de 15 anos de experiência na execução de contêineres em ambientes de produção no Google. Durante esse tempo, ele acumulou conhecimento valioso sobre como gerenciar contêineres de maneira eficaz e escalável. No que diz respeito à escalabilidade, o Kubernetes foi projetado seguindo princípios que permitiram ao Google executar milhares de contêineres a cada semana. Uma característica notável é que ele pode ser dimensionado sem a necessidade de aumentar significativamente a equipe de operações. Isso se traduz em uma capacidade para lidar com o crescimento das demandas de aplicativos sem requerer um aumento proporcional na força de trabalho de gerenciamento. Portanto, assim como o Google conseguiu executar uma grande quantidade de contêineres de maneira eficiente, o Kubernetes possibilita essa escalabilidade semelhante em outras instâncias.

O Kubernetes atua como uma ferramenta de abordagem versátil e adaptável. Ele se encaixa tanto em cenários locais, onde você pode usar seu próprio cluster, quanto em ambientes de nuvem pública, onde ele pode ser implementado de maneira consistente. Sua natureza de código aberto

oferece a liberdade de escolher a plataforma que melhor se adequa às suas necessidades. Essa portabilidade permite que você mova seus aplicativos entre diferentes ambientes com facilidade, oferecendo uma flexibilidade considerável na implantação.

O Kubernetes oferece a capacidade de automatizar o gerenciamento de armazenamento, independentemente da origem. Isso engloba armazenamento local, opções fornecidas por provedores de nuvem pública, como GCP ou AWS, bem como soluções de armazenamento em rede, como NFS, iSCSI, Gluster, Ceph, Cinder e Flocker. A capacidade de montagem automática de armazenamento é um facilitador chave na implantação e no gerenciamento de aplicativos.

No que diz respeito ao armazenamento, o Kubernetes simplifica a montagem automática de armazenamento, independentemente de ser local, de nuvem pública (como GCP ou AWS) ou de rede (NFS, iSCSI, Gluster, Ceph, Cinder e Flocker). Isso contribui para a eficácia na implantação de aplicativos.

Em relação à segurança, o Kubernetes possibilita a criação e a atualização segura de "Secrets" e configurações de aplicativos. Isso é realizado sem a necessidade de reconstruir a imagem do contêiner ou expor informações confidenciais nas configurações. Esse recurso é vital para proteger dados sensíveis, como credenciais.

Além disso, uma característica distintiva do Kubernetes é seu empacotamento automático. Esse recurso automatiza a distribuição de contêineres com base em requisitos de recursos e restrições específicas. Isso garante um equilíbrio entre disponibilidade e eficiência, permitindo que cargas de trabalho críticas e menos críticas compartilhem recursos de forma otimizada. Isso resulta em uma utilização mais eficaz e na redução do desperdício de recursos.

Por que você precisa do Kubernetes e o que ele pode fazer

Os contêineres são uma boa maneira de agrupar e executar suas aplicações. Em um ambiente de produção, você precisa gerenciar os contêineres que executam as aplicações e garantir que não haja tempo de inatividade. Por exemplo, se um contêiner cair, outro contêiner precisa ser iniciado. Não seria mais fácil se esse comportamento fosse controlado por um sistema?

É assim que o Kubernetes salva o dia ao simplificar o gerenciamento de contêineres e garantir confiabilidade! O Kubernetes oferece uma estrutura para executar sistemas distribuídos de forma resiliente. Ele cuida do escalonamento

e da recuperação à falha de sua aplicação, fornece padrões de implantação e muito mais. Por exemplo, o Kubernetes pode gerenciar facilmente uma implantação no método canário para seu sistema.

Os principais benefícios do Kubernetes

Kubernetes atua como um orquestrador de contêineres, gerenciando a implantação, escalabilidade e operação de aplicações contidas em contêineres. Ele lida com tarefas como distribuição de contêineres em máquinas físicas ou virtuais, balanceamento de carga, monitoramento, recuperação de falhas e escalabilidade automática. Ele garante que os contêineres certos estejam sendo executados nos nós apropriados, monitorando sua saúde e tomando ações para manter o estado desejado do aplicativo.

Além disso, Kubernetes permite a implantação e gerenciamento de aplicativos em diferentes ambientes, incluindo nuvens públicas, nuvens privadas e data centers locais. Isso ajuda a alcançar portabilidade de aplicativos, permitindo que você execute suas aplicações em qualquer ambiente de sua escolha.

Por fim, Kubernetes pode ser integrado a várias ferramentas de desenvolvimento e operações (DevOps), como Docker (para criação e empacotamento de contêineres) e Jenkins (para automação de integração contínua e implantação contínua). Essa integração ajuda a criar pipelines de entrega contínua eficientes e automatizados.

Funcionalidade	Descrição
Descoberta de serviço e balanceamento de carga	O Kubernetes pode expor um contêiner usando o nome DNS ou seu próprio endereço IP. Se o tráfego para um contêiner for alto, o Kubernetes pode balancear a carga e distribuir o tráfego de rede para que a implantação seja estável.
Orquestração de armazenamento	O Kubernetes permite que você monte automaticamente um sistema de armazenamento de sua escolha, como armazenamentos locais, provedores de nuvem pública e muito mais.
Lançamentos e reversões automatizadas	Você pode descrever o estado desejado para seus contêineres implantados usando o Kubernetes, e ele pode alterar o estado real para o estado desejado em um ritmo controlado. Por exemplo, você pode automatizar o Kubernetes para criar novos contêineres para sua implantação, remover os contêineres existentes e adotar todos os seus recursos para o novo contêiner.

Empacotamento binário automático	Você fornece ao Kubernetes um cluster de nós que pode ser usado para executar tarefas nos contêineres. Você informa ao Kubernetes de quanta CPU e memória (RAM) cada contêiner precisa. O Kubernetes pode encaixar contêineres em seus nós para fazer o melhor uso de seus recursos.
Autocorreção	O Kubernetes reinicia os contêineres que falham, substitui os contêineres, elimina os contêineres que não respondem à verificação de integridade definida pelo usuário e não os anuncia aos clientes até que estejam prontos para servir.
Gerenciamento de configuração e de segredos	O Kubernetes permite armazenar e gerenciar informações confidenciais, como senhas, tokens OAuth e chaves SSH. Você pode implantar e atualizar segredos e configuração de aplicações sem reconstruir suas imagens de contêiner e sem expor segredos em sua pilha de configuração.

(CESPE – TRT - 8ª Região (PA e AP) – 2022) Entre as funcionalidades do Kubernetes, a que gerencia os contêineres quanto à verificação de integridade definida pelo usuário é

a) gerenciamento de configuração e de segredos.
b) orquestração de armazenamento.
c) lançamentos e reversões automatizadas.
d) empacotamento binário automático.
e) autocorreção.

Comentários:

A funcionalidade do Kubernetes que gerencia os contêineres quanto à verificação de integridade definida pelo usuário é a opção e) autocorreção.

O recurso de autocorreção no Kubernetes é responsável por monitorar o estado dos Pods e dos contêineres em um cluster. Caso um Pod ou contêiner falhe ou não responda às verificações de integridade definidas pelo usuário, o Kubernetes tomará ações para corrigir automaticamente a situação. Isso pode incluir o reinício de contêineres com falha, a substituição de contêineres com problemas e até mesmo a eliminação de Pods não saudáveis. Portanto, a resposta correta é a

opção e) autocorreção. Esse recurso contribui para a alta disponibilidade e confiabilidade dos aplicativos implantados no Kubernetes. (Gabarito: Letra E).

O Kubernetes não é um sistema PaaS (plataforma como serviço) tradicional e completo. Como o Kubernetes opera no nível do contêiner, e não no nível do hardware, ele fornece alguns recursos geralmente aplicáveis comuns às ofertas de PaaS, como implantação, escalonamento, balanceamento de carga, e permite que os usuários integrem suas soluções de logging, monitoramento e alerta. No entanto, o Kubernetes não é monolítico, e essas soluções padrão são opcionais e conectáveis.

O Kubernetes não se encaixa em nenhuma das categorias tradicionais de serviços em nuvem (IaaS, PaaS ou CaaS, etc). Ele é uma plataforma de nível intermediário que fornece alguns recursos de IaaS, PaaS e CaaS. Por exemplo, o Kubernetes fornece recursos de IaaS, como a capacidade de criar e gerenciar máquinas virtuais. Ele também fornece recursos de PaaS, como a capacidade de implantar e gerenciar aplicativos em contêineres. E ele fornece recursos de CaaS, como a capacidade de fornecer uma plataforma para desenvolvedores criarem e implantarem aplicativos em contêineres.

(CESPE – BANRISUL – 2022) Em relação a contêineres em aplicações, julgue o item a seguir.

O Kubernetes faz o escalonamento e a recuperação no caso de falha de uma aplicação.

Comentários:

O Kubernetes é uma plataforma de orquestração de contêineres que abrange funcionalidades essenciais, como escalonamento automático e recuperação em caso de falha de aplicativos.

No que diz respeito ao Escalonamento Automático, o Kubernetes oferece a possibilidade de definir políticas de escalonamento automático para os Pods de um aplicativo. Isso implica que, baseado em métricas de uso de recursos, como CPU ou memória, o Kubernetes pode ajustar automaticamente o número de réplicas (instâncias) de um Pod. Esse mecanismo permite que a aplicação se adapte às flutuações de demanda, aumentando ou diminuindo recursos conforme necessário. Isso resulta em uma alocação de recursos mais eficiente e ajuda a garantir que o aplicativo possa enfrentar cargas elevadas sem comprometer sua performance.

Além disso, o Kubernetes também é capaz de realizar a Recuperação de Falhas de forma automática e proativa. A plataforma monitora constantemente o estado dos Pods e dos contêineres em um cluster. Se um Pod ou contêiner apresentar falhas, o Kubernetes age imediatamente para reiniciar o contêiner ou substituir o Pod defeituoso. Isso significa que o aplicativo pode se manter operacional mesmo diante de imprevistos, minimizando o tempo de inatividade. Esse aspecto é crucial para manter a alta disponibilidade dos serviços, assegurando que o aplicativo continue funcionando conforme o planejado, mesmo em cenários de falhas.

Assim, a afirmação está correta ao enfatizar que o Kubernetes oferece funcionalidades de escalonamento automático e recuperação de falhas, contribuindo para a sustentação da disponibilidade e do desempenho dos aplicativos hospedados no cluster. (Gabarito: Correto).

Orquestração de contêineres prontos para produção

Kubernetes (K8s) é um produto Open Source utilizado para automatizar a implantação, o dimensionamento e o gerenciamento de aplicativos em contêiner.

Ele agrupa contêineres que compõem uma aplicação em unidades lógicas para facilitar o gerenciamento e a descoberta de serviço. O Kubernetes se baseia em 15 anos de experiência na execução de containers em produção no Google, combinado com as melhores ideias e práticas da comunidade.

Concebido com base nos mesmos princípios que permitem ao Google executar milhares de contêineres por semana, o Kubernetes pode ser redimensionado sem aumentar sua equipe de operações.

Seja testando localmente ou executando uma empresa global, a flexibilidade do Kubernetes cresce com você para fornecer seus aplicativos de maneira consistente e fácil, independentemente de quão complexa seja sua necessidade.

O Kubernetes é Open Source, o que te oferece a liberdade de utilizá-lo em seu próprio cluster local, em uma arquitetura híbrida ou em qualquer Provedor de Computação na Nuvem público, permitindo que você mova sem esforço suas aplicações para onde você quiser.

Objetos no Kubernetes

Compreendendo os objetos no Kubernetes

Objetos no Kubernetes são entidades persistentes no sistema Kubernetes. O Kubernetes utiliza essas entidades para representar o estado do seu cluster. Especificamente, eles podem descrever:

- Quais aplicações em contêiner estão em execução (e em quais nós)
- Os recursos disponíveis para essas aplicações
- As políticas relacionadas ao comportamento dessas aplicações, como políticas de reinicialização, atualizações e tolerância a falhas

Um objeto no Kubernetes é um "registro de intenção" - uma vez que você cria o objeto, o sistema Kubernetes trabalhará constantemente para garantir que o objeto exista. Ao criar um objeto, você está efetivamente dizendo ao sistema Kubernetes como você deseja que a carga de trabalho do seu cluster pareça; este é o estado desejado do seu cluster.

Para trabalhar com objetos no Kubernetes - seja para criá-los, modificá-los ou excluí-los - você precisará usar a API do Kubernetes. Quando você usa a interface de linha de comando kubectl[1], por exemplo, a CLI faz as chamadas necessárias para a API do Kubernetes para você. Você também pode usar a API do Kubernetes diretamente em seus próprios programas usando uma das Bibliotecas de Clientes.

Especificação do objeto e status

Quase todo objeto no Kubernetes inclui dois campos de objeto aninhados que governam a configuração do objeto: a especificação do objeto e o status do objeto. Para objetos que têm uma especificação, você precisa defini-la ao criar o objeto, fornecendo uma descrição das características que deseja que o recurso tenha: seu estado desejado.

[1] A ferramenta de linha de comando kubectl suporta várias maneiras diferentes de criar e gerenciar objetos no Kubernetes.

O status descreve o estado atual do objeto, fornecido e atualizado pelo sistema Kubernetes e seus componentes. O plano de controle do Kubernetes gerencia continuamente e ativamente o estado atual de cada objeto para corresponder ao estado desejado fornecido por você.

Por exemplo: no Kubernetes, um Deployment é um objeto que pode representar uma aplicação em execução no seu cluster. Ao criar o Deployment, você pode definir a especificação do Deployment para especificar que deseja que três réplicas da aplicação estejam em execução. O sistema Kubernetes lê a especificação do Deployment e inicia três instâncias da aplicação desejada - atualizando o status para corresponder à sua especificação. Se alguma dessas instâncias falhar (uma mudança de status), o sistema Kubernetes responde à diferença entre a especificação e o status fazendo uma correção - neste caso, iniciando uma instância de substituição.

API do Kubernetes

A API do Kubernetes atua como o centro central para controlar as funcionalidades do plano de controle do Kubernetes. Ela opera por meio do servidor de API, que expõe uma API HTTP. Essa API permite que várias partes, incluindo os usuários finais, diferentes componentes dentro do seu cluster e entidades externas, estabeleçam comunicação e interação.

Por meio da API do Kubernetes, você adquire a capacidade de tanto questionar quanto gerenciar o status de objetos da API no ecossistema do Kubernetes. Esses objetos abrangem várias entidades, como Pods, Namespaces, ConfigMaps e Eventos. Embora muitas ações possam ser executadas por meio da interface de linha de comando kubectl ou ferramentas similares como kubeadm, que, por sua vez, interagem com a API, o acesso direto à API também é viável por meio de chamadas REST.

Se você está desenvolvendo um aplicativo que envolve a utilização da API do Kubernetes, é aconselhável explorar o uso de uma das bibliotecas de cliente disponíveis. Essas bibliotecas facilitam a interação contínua com a API dentro do contexto do seu aplicativo.

Grupos de API e Versionamento

Para tornar mais fácil a eliminação de campos ou a reestruturação de representações de recursos, o Kubernetes suporta múltiplas versões de API, cada uma em um caminho de API diferente, como /api/v1 ou /apis/rbac.authorization.k8s.io/v1alpha1.

O versionamento é realizado no nível da API, em vez de no nível do recurso ou campo, para garantir que a API apresente uma visão clara e consistente dos recursos e comportamentos do sistema, e para permitir o controle de acesso a APIs em fim de vida útil e/ou experimentais.

Para facilitar a evolução e a extensão da sua API, o Kubernetes implementa grupos de API que podem ser habilitados ou desabilitados.

Os recursos da API são distinguíveis por seu grupo de API, tipo de recurso, espaço de nomes (para recursos com espaços de nomes) e nome. O servidor de API trata a conversão entre versões de API de forma transparente: todas as diferentes versões são, na verdade, representações dos mesmos dados persistidos. O servidor de API pode servir os mesmos dados subjacentes por meio de múltiplas versões de API.

Por exemplo, suponha que existam duas versões de API, v1 e v1beta1, para o mesmo recurso. Se você originalmente criou um objeto usando a versão v1beta1 de sua API, você pode posteriormente ler, atualizar ou excluir esse objeto usando a versão v1beta1 ou a versão v1 da

API, até que a versão v1beta1 seja descontinuada e removida. Nesse ponto, você pode continuar acessando e modificando o objeto usando a versão v1 da API.

Arquitetura do Cluster

Um cluster Kubernetes é um conjunto de máquinas que colaboram para executar aplicações em containers. Essas máquinas, conhecidas como nós, são divididas em duas categorias: nós mestres e nós trabalhadores. Os nós mestres são responsáveis por gerenciar o cluster, alocar recursos para os nós e garantir a saúde das aplicações em execução. Já os nós trabalhadores são onde as aplicações em containers são executadas, fornecendo os recursos como CPU, memória e armazenamento necessários para a execução dessas aplicações.

Um cluster Kubernetes pode ser composto por qualquer número de nós, mas geralmente consiste em pelo menos três para garantir a operação contínua, mesmo em caso de falha de um ou mais nós. Ao utilizar o Kubernetes, você está implantando um cluster que pode ser hospedado em suas próprias máquinas ou em ambientes de nuvem.

Os benefícios do uso do Kubernetes incluem escalabilidade, garantia de disponibilidade, recursos de segurança, facilidade de gerenciamento e automação de implantação, dimensionamento e atualização de aplicações. Ele fornece uma infraestrutura robusta para orquestrar e gerenciar ambientes complexos de aplicações em containers.

Nodes ou Nós

O Kubernetes executa sua carga de trabalho ao colocar contêineres em Pods para serem executados em Nós. Um nó pode ser uma máquina virtual ou física, dependendo do cluster. Cada nó é gerenciado pelo plano de controle e contém os serviços necessários para executar Pods.

Normalmente, você tem vários nós em um cluster; em um ambiente de aprendizado ou com recursos limitados, você pode ter apenas um nó. Os componentes em um nó incluem o kubelet, um tempo de execução de contêiner e o kube-proxy. Mas o que é um kublet, professora?

O kubelet é o principal "agente de nó" que é executado em cada nó. Ele pode registrar o nó no servidor de API usando um dos seguintes métodos: o nome do host; uma sinalização para substituir o nome do host; ou lógica específica para um provedor de nuvem.

O kubelet opera com base em uma PodSpec. Uma PodSpec é um objeto YAML ou JSON que descreve um pod. O kubelet recebe um conjunto de PodSpecs fornecidos por meio de vários mecanismos (principalmente através do servidor de API) e garante que os contêineres descritos nesses PodSpecs estejam em execução e saudáveis. O kubelet não gerencia contêineres que não foram criados pelo Kubernetes.

Gerenciamento

Existem duas maneiras principais de adicionar Nós ao servidor de API:

- O kubelet em um nó se registra automaticamente no plano de controle
- Você adiciona manualmente um objeto Node

Após criar um objeto Node, ou o kubelet em um nó se registrar automaticamente, o plano de controle verifica se o novo objeto Node é válido. Por exemplo, se você tentar criar um Node a partir do seguinte manifesto JSON

```
{
  "kind": "Node",
  "apiVersion": "v1",
  "metadata": {
    "name": "10.240.79.157",
    "labels": {
      "name": "my-first-k8s-node"
    }
  }
}
```

O Kubernetes cria internamente um objeto Node (a representação). O Kubernetes verifica se um kubelet se registrou no servidor de API que corresponde ao campo metadata.name do Node. Se o nó estiver saudável (ou seja, todos os serviços necessários estão em execução), ele estará apto para executar um Pod. Caso contrário, esse nó é ignorado para qualquer atividade do cluster até que se torne saudável.

O nome identifica um Node. Dois Nodes não podem ter o mesmo nome ao mesmo tempo. O Kubernetes também assume que um recurso com o mesmo nome é o mesmo objeto. No caso de um Node, é implicitamente assumido que uma instância que usa o mesmo nome terá o mesmo estado (por exemplo, configurações de rede, conteúdo do disco raiz) e atributos como rótulos do nó. Isso pode levar a inconsistências se uma instância for modificada sem alterar seu nome. Se o Node precisar ser substituído ou atualizado significativamente, o objeto Node existente precisa ser removido primeiro do servidor de API e recriado após a atualização.

Status do Nó

O status de um nó contém as seguintes informações:

- Endereços
- Condições

- Capacidade e Alocável
- Informações

Você pode usar o kubectl para visualizar o status de um nó e outros detalhes:

kubectl describe node <insira-o-nome-do-nó-aqui>

Endereços

O uso desses campos varia dependendo do seu provedor de nuvem ou configuração de metal nu.

- HostName: O nome do host conforme reportado pelo kernel do nó. Pode ser substituído através do parâmetro --hostname-override do kubelet.
- ExternalIP: Geralmente, o endereço IP do nó que é acessível externamente (disponível fora do cluster).
- InternalIP: Geralmente, o endereço IP do nó que é acessível apenas dentro do cluster.

Condições

O campo de condições descreve o estado de todos os nós em execução. Exemplos de condições incluem:

Condição do Nó	Descrição
Ready	Verdadeiro se o nó estiver saudável e pronto para aceitar pods, Falso se o nó não estiver saudável e não estiver aceitando pods, e Desconhecido se o controlador do nó não tiver recebido informações do nó no último período de graça do monitor de nós (padrão é 40 segundos)
DiskPressure	Verdadeiro se houver pressão sobre o tamanho do disco – ou seja, se a capacidade do disco estiver baixa; caso contrário, Falso
MemoryPressure	Verdadeiro se houver pressão sobre a memória do nó – ou seja, se a memória do nó estiver baixa; caso contrário, Falso
PIDPressure	Verdadeiro se houver pressão sobre os processos – ou seja, se houver muitos processos no nó; caso contrário, Falso

NetworkUnavailable	Verdadeiro se a rede para o nó não estiver configurada corretamente, caso contrário, Falso

Na API do Kubernetes, a condição de um nó é representada como parte do .status do recurso Node. Por exemplo, a seguinte estrutura JSON descreve um nó saudável:

```
"conditions": [
{
"type": "Ready",
"status": "True",
"reason": "KubeletReady",
"message": "kubelet está enviando o status de pronto",
"lastHeartbeatTime": "2019-06-05T18:38:35Z",
"lastTransitionTime": "2019-06-05T11:41:27Z"
}
]
```

Quando ocorrem problemas nos nós, o plano de controle do Kubernetes cria automaticamente manchas que correspondem às condições que afetam o nó. Um exemplo disso é quando o status da condição Ready permanece Desconhecido ou Falso por mais tempo do que o período de graça de monitoramento de nó do kube-controller-manager, que é 40 segundos por padrão. Isso causará a adição de uma mancha node.kubernetes.io/unreachable para um status Desconhecido, ou uma mancha node.kubernetes.io/not-ready para um status Falso, ao Nó.

Essas manchas afetam os pods pendentes, já que o agendador leva em consideração as manchas do Nó ao atribuir um pod a um Nó. Os pods existentes agendados para o nó podem ser evacuados devido à aplicação de manchas NoExecute. Os pods também podem ter tolerações que permitem que eles sejam agendados e continuem em execução em um Nó, mesmo que ele tenha uma mancha específica.

Capacidade e alocável

Descreve os recursos disponíveis no nó: CPU, memória e o número máximo de pods que podem ser agendados no nó.

Os campos do bloco de capacidade indicam a quantidade total de recursos que um Node possui. O bloco alocável indica a quantidade de recursos em um Node que está disponível para ser consumido por Pods normais.

Controlador de nó

O controlador do nó é um componente do plano de controle do Kubernetes que gerencia vários aspectos dos nós. O controlador de nó tem várias funções na vida de um nó. A primeira é atribuir um bloco CIDR ao nó quando ele é registrado (se a atribuição CIDR estiver ativada).

A segunda é manter a lista interna de nós do controlador de nó atualizada com a lista de máquinas disponíveis do provedor de nuvem. Ao executar em um ambiente de nuvem e sempre que um nó não estiver íntegro, o controlador do nó pergunta ao provedor de nuvem se a VM desse nó ainda está disponível. Caso contrário, o controlador de nó exclui o nó de sua lista de nós. A terceira é monitorar a integridade dos nós. O controlador de nó é responsável por:

Caso um nó fique inacessível, atualize a condição Ready no campo status do Nó. Nesse caso, o controlador do nó define a condição Ready como Unknown.

Se um nó permanecer inacessível: acionar a remoção iniciada pela API para todos os pods no nó inacessível. Por padrão, o controlador do nó espera 5 minutos entre marcar o nó como Unknowne enviar a primeira solicitação de despejo.

Por padrão, o controlador de nó verifica o estado de cada nó a cada 5 segundos. Este período pode ser configurado através do --node-monitor-periodsinalizador no kube-controller-manager componente.

Controladores

Para entender o conceito de controladores na tecnologia, podemos recorrer a uma analogia comum na vida cotidiana: o termostato de um quarto. Imagine que você está em um ambiente e deseja que a temperatura fique em um certo nível de conforto, digamos 22°C. Você configura o termostato para essa temperatura desejada. No entanto, a temperatura real do ambiente pode variar devido a condições externas.

O termostato é como um controlador nesse cenário. Ele monitora constantemente a temperatura atual do ambiente, que é o estado em que as coisas estão. A temperatura desejada que você configurou é o estado que você quer alcançar. O termostato age como um regulador para aproximar o estado atual do estado desejado.

Se a temperatura atual estiver abaixo dos 22°C, o termostato liga o aquecedor para aquecer o ambiente. À medida que a temperatura sobe e se aproxima dos 22°C, o termostato desliga o aquecedor para evitar um superaquecimento. Isso cria um ciclo contínuo de monitoramento e ação para manter a temperatura em um nível confortável.

No mundo da TI, os controladores são como esses termostatos virtuais. Eles observam constantemente o estado do seu "quarto virtual", que é o cluster de servidores e aplicações. Assim

como o termostato, os controladores buscam manter o estado do cluster o mais próximo possível do estado desejado.

Por exemplo, um controlador no Kubernetes pode monitorar o número de réplicas de um aplicativo que você deseja manter em execução. Se por algum motivo o número de réplicas cair abaixo do desejado, o controlador agirá, iniciando novas réplicas para restaurar o equilíbrio. Se houver réplicas extras que não são necessárias, o controlador pode escalar para baixo, desligando algumas delas.

Um controlador rastreia pelo menos um tipo de recurso do Kubernetes. Esses objetos possuem um campo de especificação que representa o estado desejado. Os controladores desse recurso são responsáveis por fazer o estado atual se aproximar desse estado desejado. O controlador pode executar a ação por si só; mais comumente, no Kubernetes, um controlador enviará mensagens para o servidor da API que têm efeitos colaterais úteis. Você verá exemplos disso abaixo.

Controle via servidor da API

O controlador Job é um exemplo de um controlador embutido no Kubernetes. Os controladores embutidos gerenciam o estado interagindo com o servidor da API do cluster. Job é um recurso do Kubernetes que executa um Pod, ou talvez vários Pods, para realizar uma tarefa e, em seguida, para. Uma vez agendados, os objetos Pod se tornam parte do estado desejado para um kubelet.

Quando o controlador Job identifica uma nova tarefa, ele garante que, em algum lugar do seu cluster, os kubelets em um conjunto de Nós estejam executando o número correto de Pods para realizar o trabalho. O controlador Job não executa nenhum Pod ou contêiner por si só. Em vez disso, o controlador Job informa ao servidor da API para criar ou remover Pods. Outros componentes no plano de controle atuam nas novas informações (há novos Pods para agendar e executar) e, eventualmente, o trabalho é concluído.

Depois de criar um novo Job, o estado desejado é que esse Job seja concluído. O controlador Job faz com que o estado atual desse Job esteja mais próximo do seu estado desejado: criando Pods que realizam o trabalho desejado para esse Job, para que o Job esteja mais próximo da conclusão.

Os controladores também atualizam os objetos que os configuram. Por exemplo: após a conclusão do trabalho de um Job, o controlador Job atualiza esse objeto Job para marcá-lo como Concluído. Isso é como alguns termostatos desligam uma luz para indicar que o quarto está agora na temperatura que você definiu.

Controle direto

Em contraste com o Job, alguns controladores precisam fazer alterações em coisas fora do seu cluster. Por exemplo, se você usa um laço de controle para garantir que haja Nodes suficientes no seu cluster, então esse controlador precisa de algo fora do cluster atual para configurar novos Nodes quando necessário.

Controladores que interagem com estado externo encontram seu estado desejado a partir do servidor da API, e depois se comunicam diretamente com um sistema externo para aproximar o estado atual em conformidade. De fato, há um controlador que dimensiona horizontalmente os nós no seu cluster.

O ponto importante aqui é que o controlador faz algumas alterações para atingir o estado desejado e, em seguida, relata o estado atual de volta ao servidor da API do seu cluster. Outros laços de controle podem observar esses dados relatados e tomar suas próprias ações.

No exemplo do termostato, se o quarto estiver muito frio, um controlador diferente também poderá ligar um aquecedor de proteção contra geada. Com clusters Kubernetes, o plano de controle trabalha indiretamente com ferramentas de gerenciamento de endereços IP, serviços de armazenamento, APIs de provedores de nuvem e outros serviços, estendendo o Kubernetes para implementar isso.

Contexto

No contexto do Kubernetes, um **context** ou "**contexto**" se refere a um conjunto de configurações que determinam como o cliente (como o utilitário de linha de comando kubectl) se conecta a um cluster Kubernetes e a um usuário específico. Isso permite que você gerencie várias configurações de acesso a diferentes clusters e contextos dentro do ecossistema do Kubernetes.

Containers

Cada contêiner que você executa é repetível; a padronização ao incluir as dependências significa que você obtém o mesmo comportamento onde quer que o execute. Os contêineres separam os aplicativos da infraestrutura do host subjacente. Isso torna a implantação mais fácil em diferentes ambientes de nuvem ou sistemas operacionais.

Cada nó em um cluster Kubernetes executa os contêineres que compõem os Pods atribuídos a esse nó. Os contêineres em um Pod são co-localizados e co-agendados para serem executados no mesmo nó.

Uma imagem de contêiner é um pacote de software pronto para ser executado, contendo tudo o que é necessário para rodar um aplicativo: o código e qualquer runtime que ele requer, bibliotecas de aplicativos e de sistema, e valores padrão para quaisquer configurações essenciais.

Contêineres têm a intenção de serem stateless e imutáveis: você não deve alterar o código de um contêiner que já está em execução. Se você tem um aplicativo em contêiner e deseja fazer alterações, o processo correto é construir uma nova imagem que inclua a mudança e, em seguida, recriar o contêiner para iniciar a partir da imagem atualizada.

O runtime de contêiner é o software responsável por executar os contêineres. O Kubernetes suporta runtimes de contêiner como containerd, CRI-O e qualquer outra implementação da CRI (Interface de Runtime de Contêiner) do Kubernetes.

Normalmente, você pode permitir que o cluster escolha o runtime de contêiner padrão para um Pod. Se você precisar usar mais de um runtime de contêiner em seu cluster, pode especificar a RuntimeClass para um Pod para garantir que o Kubernetes execute esses contêineres usando um runtime de contêiner específico.

Você também pode usar RuntimeClass para executar diferentes Pods com o mesmo runtime de contêiner, mas com configurações diferentes.

Imagens

Uma imagem de contêiner representa dados binários que encapsulam um aplicativo e todas as suas dependências de software. As imagens de contêiner são pacotes de software executáveis que podem ser executados de forma independente e que fazem suposições muito bem definidas sobre o ambiente de execução.

Normalmente, você cria uma imagem de contêiner para o seu aplicativo e a envia para um registro antes de fazer referência a ela em um Pod.

Nomes de Imagem

As imagens de contêiner normalmente recebem um nome, como pause, example/mycontainer ou kube-apiserver. As imagens também podem incluir um nome de host do registro; por exemplo: fictional.registry.example/imagename, e possivelmente um número de porta também; por exemplo: fictional.registry.example:10443/imagename.

Se você não especificar um nome de host do registro, o Kubernetes assume que você está se referindo ao registro público do Docker. Após a parte do nome da imagem, você pode adicionar uma tag (da mesma forma que faria ao usar comandos como docker ou podman). As tags permitem identificar diferentes versões da mesma série de imagens.

As tags de imagem consistem em letras maiúsculas e minúsculas, dígitos, sublinhados (), pontos (.) e traços (-). Existem regras adicionais sobre onde você pode colocar os caracteres separadores (, -, e .) dentro de uma tag de imagem. Se você não especificar uma tag, o Kubernetes assume que você quer dizer a tag latest.

Atualizando Imagens

Quando você cria pela primeira vez um Deployment, StatefulSet, Pod ou outro objeto que inclui um modelo de Pod, por padrão, a política de busca de todos os contêineres nesse pod será definida como IfNotPresent, se não for especificada explicitamente. Essa política faz com que o kubelet ignore a busca de uma imagem se ela já existir.

Política de Busca de Imagem

A imagePullPolicy para um contêiner e a tag da imagem afetam quando o kubelet tenta buscar (baixar) a imagem especificada.

Vejamos uma lista dos valores que você pode definir para imagePullPolicy e os efeitos desses valores:

Funcionalidade	Descrição
IfNotPresent	A imagem é buscada apenas se ainda não estiver presente localmente.
Always	Sempre que o kubelet inicia um contêiner, o kubelet consulta o registro da imagem do contêiner para resolver o nome em um digest de imagem. Se o kubelet tiver uma imagem de contêiner com esse digest exato armazenado em cache localmente, o kubelet usa a imagem em cache;

	caso contrário, o kubelet busca a imagem com o digest resolvido e usa essa imagem para iniciar o contêiner.
Never	O kubelet não tenta buscar a imagem. Se a imagem já estiver presente localmente de alguma forma, o kubelet tenta iniciar o contêiner; caso contrário, a inicialização falha. Consulte imagens pré-baixadas para mais detalhes.

A semântica de cache do provedor de imagem subjacente torna até mesmo imagePullPolicy: Always eficiente, desde que o registro seja acessível de forma confiável. Seu runtime de contêiner pode perceber que as camadas de imagem já existem no nó, para que elas não precisem ser baixadas novamente.

Para garantir que o Pod sempre use a mesma versão de uma imagem de contêiner, você pode especificar o digest da imagem; substitua <image-name>:<tag> por <image-name>@<digest> (por exemplo, image@sha256:45b23dee08af5e43a7fea6c4cf9c25ccf269ee113168c19722f87876677c5cb2).

Ao usar tags de imagem, se o registro da imagem mudar o código que a tag da imagem representa, você pode acabar com uma mistura de Pods executando o código antigo e novo. Um digest de imagem identifica de forma única uma versão específica da imagem, então o Kubernetes executa o mesmo código toda vez que inicia um contêiner com esse nome e digest de imagem especificados. Especificar uma imagem por digest corrige o código que você executa, para que uma mudança no registro não possa levar a essa mistura de versões.

Política de busca de imagem padrão

A política de busca de imagem determina como o Kubernetes deve agir ao buscar uma imagem de contêiner de um registro. No entanto, se você não especificar essa política explicitamente, o Kubernetes tomará decisões com base em certas situações:

1. Se você não definir explicitamente imagePullPolicy e especificar o "digest" da imagem do contêiner, o Kubernetes definirá automaticamente a política como IfNotPresent. Isso significa que o Kubernetes puxará a imagem somente se ela não estiver presente localmente, garantindo que a imagem existente seja reutilizada sempre que possível..

2. Se você omitir a definição da imagePullPolicy e a tag da imagem do contêiner for :latest, o Kubernetes definirá automaticamente a política como Always. Isso significa que o Kubernetes sempre puxará a imagem mais recente, mesmo que ela já esteja presente

localmente. Isso é útil quando você deseja garantir que a versão mais atualizada da imagem seja usada.

3. Quando você não definir explicitamente imagePullPolicy e não especificar a tag da imagem do contêiner, o Kubernetes definirá automaticamente a política como Always. Isso garante que a imagem seja sempre puxada, independentemente da tag, o que pode ser útil para garantir que você tenha a versão mais recente disponível.

4. Caso você omita a definição da imagePullPolicy e especifique uma tag para a imagem do contêiner que não seja :latest, o Kubernetes definirá automaticamente a política como IfNotPresent. Isso significa que a imagem será puxada apenas se não estiver presente localmente, visando reutilizar a imagem existente.

Para tentar ajudar a entender, aqui está a tabela explicativa das políticas de busca de imagem no Kubernetes quando o campo imagePullPolicy é omitido:

imagePullPolicy padrão definido automaticamente	Situação
IfNotPresent	Especificar digest da imagem do contêiner
Always	Tag da imagem do contêiner é :latest
Always	Não especificar tag da imagem do contêiner
IfNotPresent	Especificar outra tag da imagem do contêiner

Essa tabela resume como o Kubernetes decide a política de busca de imagem com base nas situações específicas. Quando você não fornece explicitamente a política, o Kubernetes a define automaticamente, garantindo que a busca de imagens seja feita de maneira apropriada para cada cenário. Isso ajuda a otimizar a eficiência e o desempenho das operações do contêiner.

Requisito de Atualização de Imagem

Se você deseja garantir sempre a obtenção da imagem mais recente, você pode seguir uma das seguintes opções:

- Defina a política de obtenção de imagem (imagePullPolicy) do contêiner como Always.
- Deixe o campo imagePullPolicy em branco e use :latest como a marcação (tag) para a imagem que será usada; o Kubernetes configurará automaticamente a política como Always quando você submeter o Pod.

- Deixe o campo imagePullPolicy em branco e não especifique a marcação (tag) da imagem que será usada; o Kubernetes configurará automaticamente a política como Always quando você submeter o Pod.
- Habilite o controlador de admissão AlwaysPullImages.

Ambiente de contêineres

O ambiente de contêineres do Kubernetes fornece vários recursos importantes para Contêineres:

- Um sistema de arquivos, que é uma combinação de uma imagem e um ou mais volumes.
- Informações sobre o próprio Contêiner.
- Informações sobre outros objetos no cluster.

Informações do Contêiner.

O nome de host de um Contêiner é o nome do Pod no qual o Contêiner está sendo executado. Ele está disponível por meio do comando de nome de host (hostname) ou da chamada de função gethostname na biblioteca C (libc). O nome do Pod e o espaço de nomes estão disponíveis como variáveis de ambiente por meio da API descendente (downward API).

Variáveis de ambiente definidas pelo usuário na definição do Pod também estão disponíveis para o Contêiner, assim como quaisquer variáveis de ambiente especificadas estaticamente na imagem do contêiner.

Informações do cluster

Uma lista de todos os serviços que estavam em execução quando um Contêiner foi criado está disponível para esse Contêiner como variáveis de ambiente. Essa lista é limitada aos serviços no mesmo espaço de nomes (namespace) do Pod do novo Contêiner e aos serviços do plano de controle do Kubernetes. Para um serviço chamado "foo" que mapeia para um Contêiner chamado "bar", as seguintes variáveis são definidas:

```
FOO_SERVICE_HOST=<the host the service is running on>
FOO_SERVICE_PORT=<the port the service is running on>
```

Hooks de ciclo de vida para os Contêineres

Análogo a muitos frameworks de linguagens de programação que possuem hooks no ciclo de vida dos componentes, como o Angular, o Kubernetes oferece Hooks de Ciclo de Vida para os Contêineres. Os hooks permitem que os Contêineres estejam cientes de eventos em seu ciclo de vida de gerenciamento e executem código implementado em um manipulador quando o hook de ciclo de vida correspondente é executado.

Esses "hooks" permitem que os Contêineres estejam cientes de eventos em seu ciclo de vida de gerenciamento e executem código implementado em um handler quando o "hook" correspondente é executado.

PostStart

Este hook é executado imediatamente após a criação de um contêiner. No entanto, não há garantia de que o hook será executado antes do ponto de entrada (ENTRYPOINT) do contêiner. Nenhum parâmetro é passado para o handler.

PreStop

Este hook é chamado imediatamente antes de um contêiner ser encerrado devido a uma solicitação de API ou evento de gerenciamento, como liveness/startup probe failure, preemption, resource contention e outros. Uma chamada ao Hook PreStop falha se o contêiner já estiver em um estado encerrado (terminated) ou concluído(completed), e o Hook deve ser concluído antes do sinal TERM para interromper o contêiner ser enviado. A contagem regressiva do período de carência de término da pod começa antes que o Hook PreStop seja executado, então, independentemente do resultado do handler, o contêiner eventualmente será encerrado dentro do período de carência de término da pod. Nenhum parâmetro é passado para o handler.

(FCC – TRT – 22ª Região (PI) – 2022) No Kubernetes, os contêineres gerenciados pelo kubelet podem usar a estrutura de hook do ciclo de vida do contêiner para executar código acionado por eventos durante o gerenciamento de seu ciclo de vida. Há dois tipos de hooks:

I. Este hook é executado imediatamente após um contêiner ser criado. Mas não há garantia de que o hook será executado antes do ENTRYPOINT do contêiner. Nenhum parâmetro é passado para o handler.

II. Esse hook é chamado imediatamente antes de um contêiner ser terminated devido a uma solicitação de API ou um gerenciamento de evento como liveness/startup probe failure, preemption, resource contention e outros. Uma chamada a este hook falha se o contêiner já está em um estado terminated ou completed e o hook deve ser concluído antes que o sinal TERM seja enviado para parar o contêiner. Nenhum parâmetro é passado para o handler.

Os hooks I e II são, correta e respectivamente,
a) PostStart e PreStop.
b) HookStart e HookStop.
c) PodStart e PodStop.
d) KubeStart e KubeStop.
e) KubeletStart e KubeletStop.

Comentários:

Pessoal, nosso gabarito é a Letra A. Vamos analisar os itens.
I. Este hook é executado imediatamente após um contêiner ser criado. Mas não há garantia de que o hook será executado antes do ENTRYPOINT do contêiner. Nenhum parâmetro é passado para o handler. -> Isso corresponde ao hook PostStart. É a definição exata!

II. Esse hook é chamado imediatamente antes de um contêiner ser terminado devido a uma solicitação de API ou um gerenciamento de evento como liveness/startup probe failure, preemption, resource contention e outros. Uma chamada a este hook falha se o contêiner já está em um estado terminated ou completed e o hook deve ser concluído antes que o sinal TERM seja enviado para parar o contêiner. Nenhum parâmetro é passado para o handler. -> Isso corresponde ao hook PreStop. É a definição exata!

Os hooks I e II permitem que os desenvolvedores executem ações específicas quando um contêiner é criado ou antes que ele seja terminado no Kubernetes. O hook PostStart é útil para realizar tarefas de inicialização após a criação de um contêiner, enquanto o hook PreStop permite que o contêiner execute ações finais antes de ser encerrado. Estes hooks são uma parte importante da gestão do ciclo de vida de um contêiner em um ambiente Kubernetes. (Gabarito: Letra A)

Implementações de Handler de Hooks

Contêineres podem acessar um Hook implementando e registrando um handler para esse Hook. Existem dois tipos de handler de Hooks que podem ser implementados para Contêineres:

- Exec - Executa um comando específico, como pre-stop.sh, dentro dos cgroups e namespaces do Contêiner. Os recursos consumidos pelo comando são contados contra o Contêiner.
- HTTP - Executa uma solicitação HTTP contra um ponto de extremidade específico no Contêiner.

Execução de Handler de Hooks

Quando um Hook de gerenciamento de ciclo de vida de Contêiner é chamado, o sistema de gerenciamento Kubernetes executa o handler de acordo com a ação do Hook, as ações httpGet e tcpSocket são executadas pelo processo kubelet, e exec é executado no contêiner.

As chamadas de handler de hooks são síncronas no contexto da Pod que contém o Contêiner. Isso significa que, para um Hook PostStart, o ponto de entrada do Contêiner e o Hook são acionados de forma assíncrona. No entanto, se o Hook demorar muito para ser executado ou ficar em espera, o Contêiner não poderá atingir um estado em execução.

Hooks PreStop não são executados de forma assíncrona em relação ao sinal para interromper o Contêiner; o Hook deve concluir sua execução antes que o sinal TERM possa ser enviado. Se um Hook PreStop ficar em espera durante a execução, a fase da Pod será "Terminando" e permanecerá assim até que a Pod seja encerrada após a expiração de seu "terminationGracePeriodSeconds". Este período de carência se aplica ao tempo total necessário tanto para a execução do Hook PreStop quanto para a parada normal do Contêiner. Se, por exemplo, "terminationGracePeriodSeconds" for 60, e o Hook levar 55 segundos para ser concluído, e o Contêiner levar 10 segundos para parar normalmente após receber o sinal, então o Contêiner será encerrado antes de poder parar normalmente, uma vez que "terminationGracePeriodSeconds" é menor que o tempo total (55+10) necessário para essas duas ações acontecerem.

Se um Hook PostStart ou PreStop falhar, ele encerrará o Contêiner. Os usuários devem tornar seus handler de Hooks o mais leves possível. No entanto, há casos em que comandos de longa duração fazem sentido, como ao salvar o estado antes de interromper um Contêiner.

Garantias de Entrega de Hooks

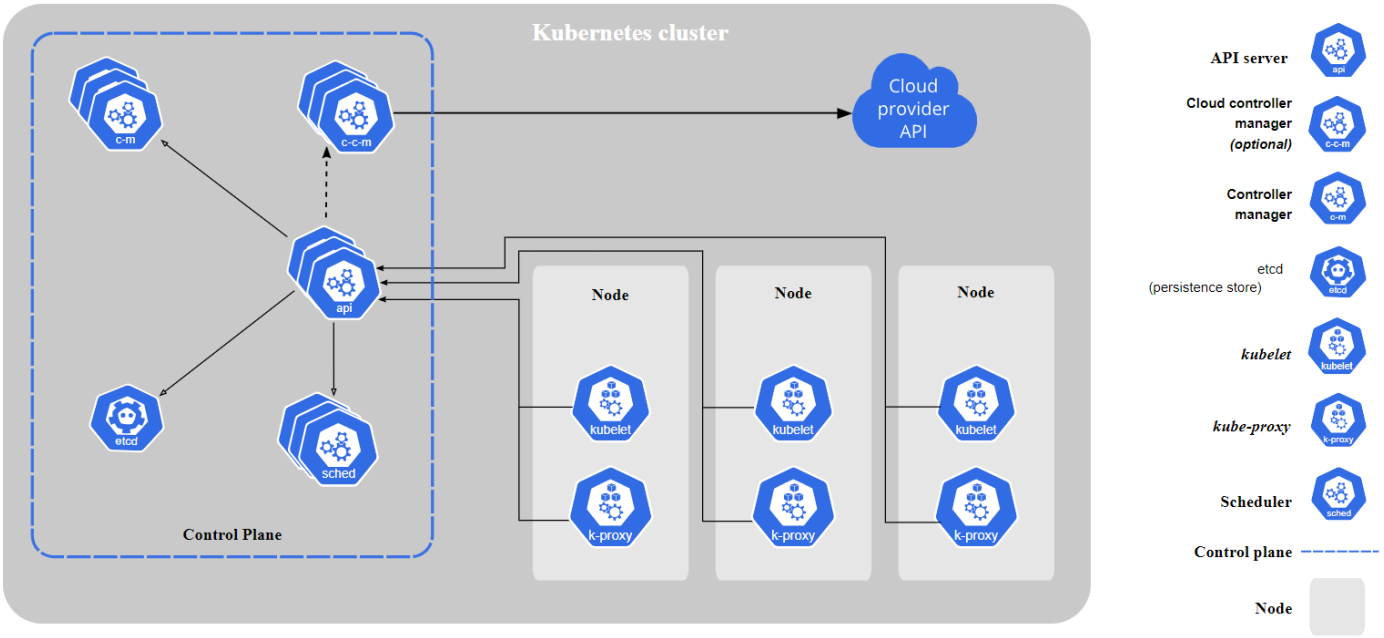
A entrega de Hooks é projetada para ser pelo menos uma vez, o que significa que um Hook pode ser chamado várias vezes para qualquer evento dado, como no caso de PostStart ou PreStop. Cabe à implementação do Hook lidar corretamente com isso.

Geralmente, são feitas apenas entregas únicas. Por exemplo, se um receptor de Hooks HTTP estiver inoperante e não puder processar tráfego, não haverá tentativa de reenvio. No entanto, em alguns casos raros, pode ocorrer a entrega dupla. Por exemplo, se um kubelet for reiniciado no meio do envio de um Hook, o Hook poderá ser reenviado depois que o kubelet voltar a funcionar.

Debugging Hook handlers

Os registros para um Hook handler não são expostos nos eventos da Pod. Se um handler falha por algum motivo, ele emite um evento. Para o "PostStart", isso é o evento "FailedPostStartHook", e para o "PreStop", isso é o evento "FailedPreStopHook". Para gerar manualmente um evento "FailedPostStartHook" com falha, modifique o arquivo "lifecycle-events.yaml" para alterar o comando "postStart" para "badcommand" e aplique-o.

Componentes Kubernetes



Quando você implanta o Kubernetes, você obtém um cluster. Um cluster Kubernetes consiste em um conjunto de máquinas de trabalho, chamadas de nós (nodes), que executam aplicações em contêineres. Cada cluster possui pelo menos um nó de trabalho.

Os nós de trabalho hospedam os Pods que são os componentes da carga de trabalho da aplicação. O plano de controle (control plane) gerencia os nós de trabalho e os Pods no cluster. Em ambientes de produção, o plano de controle geralmente é executado em várias máquinas e um cluster geralmente executa vários nós, proporcionando tolerância a falhas e alta disponibilidade.

Componentes do Plano de Controle

Os componentes do plano de controle tomam decisões globais sobre o cluster (por exemplo, escalonamento), além de detectar e responder a eventos do cluster (por exemplo, iniciar um novo pod quando o campo "replicas" de um deployment não está satisfeito).

Os componentes do plano de controle podem ser executados em qualquer máquina do cluster. No entanto, para simplicidade, os scripts de configuração normalmente iniciam todos os componentes do plano de controle na mesma máquina e não executam contêineres de usuário nesta máquina. Veja a criação de clusters altamente disponíveis com o kubeadm para um exemplo de configuração do plano de controle que é executado em várias máquinas.

kube-apiserver

O servidor de API é um componente do plano de controle do Kubernetes que expõe a API do Kubernetes. O servidor de API é a interface de entrada para o plano de controle do Kubernetes.

A implementação principal de um servidor de API do Kubernetes é o kube-apiserver. O kube-apiserver é projetado para escalar horizontalmente, ou seja, ele escala ao implantar mais instâncias. Você pode executar várias instâncias do kube-apiserver e balancear o tráfego entre essas instâncias.

etcd

O etcd é um armazenamento de chave-valor consistente e altamente disponível usado como armazenamento de apoio do Kubernetes para todos os dados do cluster. Se o seu cluster Kubernetes usa o etcd como armazenamento de apoio, certifique-se de ter um plano de backup para os dados.

kube-scheduler

Componente do plano de controle que observa por Pods recém-criados sem nó atribuído e seleciona um nó para eles serem executados. Fatores levados em consideração para decisões de escalonamento incluem: requisitos individuais e coletivos de recursos, restrições de hardware/software/política, especificações de afinidade e anti-afinidade, localidade de dados, interferência entre cargas de trabalho e prazos.

O escalonador determina quais nós são colocações válidas para cada Pod na fila de escalonamento de acordo com restrições e recursos disponíveis. Em seguida, o escalonador classifica cada Nó válido e vincula o Pod a um Nó adequado. Múltiplos escalonadores diferentes podem ser usados dentro de um cluster; o kube-scheduler é a implementação de referência.

O componente do Kubernetes que gerencia os Pods que foram criados e que ainda não possuem nenhum nó atribuído é o kube-scheduler. O kube-scheduler é responsável por tomar decisões sobre quais nós os Pods devem ser agendados com base em critérios como requisitos de recursos, afinidades, anti-afinidades e restrições. Ele faz parte do processo de agendamento do Kubernetes, garantindo que os Pods sejam alocados nos nós apropriados para melhorar a eficiência, o desempenho e o equilíbrio de carga no cluster.

O Kube-scheduler seleciona um nó ideal para executar pods recém-criados ou ainda não agendados. Como os containers nos pods - e os próprios pods - podem ter requisitos diferentes, o escalonador filtra quaisquer nós que não atendam às necessidades específicas de agendamento de um pod. Alternativamente, a API permite que você especifique um nó para um pod quando o cria, mas isso é incomum e só é feito em casos especiais.

Em um cluster, os nós que atendem aos requisitos de agendamento de um pod são chamados de nós viáveis. Se nenhum dos nós for adequado, o pod permanece não agendado até que o escalonador possa colocá-lo.

O escalonador encontra nós viáveis para um pod e, em seguida, executa um conjunto de funções para avaliar os nós viáveis e escolhe um nó com a pontuação mais alta entre os viáveis para executar o pod. O escalonador, então, notifica o servidor da API sobre essa decisão em um processo chamado de vinculação (binding).

Fatores que precisam ser levados em consideração para decisões de agendamento incluem requisitos individuais e coletivos de recursos, restrições de hardware/software/políticas, especificações de afinidade e anti-afinidade, localidade de dados, interferência entre cargas de trabalho e assim por diante.

Seleção de Nó no kube-scheduler

O kube-scheduler seleciona um nó para o pod em uma operação de 2 etapas:

- Filtragem
- Pontuação

A etapa de filtragem encontra o conjunto de nós onde é viável agendar o Pod. Por exemplo, o filtro PodFitsResources verifica se um Node candidato tem recursos disponíveis suficientes para atender às solicitações específicas de recursos de um Pod. Após esta etapa, a lista de nós contém os Nós adequados; muitas vezes, haverá mais de um. Se a lista estiver vazia, esse Pod ainda não é agendável.

Na etapa de pontuação, o escalonador classifica os nós restantes para escolher a colocação de Pod mais adequada. O escalonador atribui uma pontuação a cada nó que sobreviveu à filtragem, baseando esta pontuação nas regras de pontuação ativas.

Finalmente, o kube-scheduler atribui o Pod ao nó com a classificação mais alta. Se houver mais de um nó com pontuações iguais, o kube-scheduler seleciona um deles aleatoriamente. Existem duas maneiras suportadas de configurar o comportamento de filtragem e pontuação do escalonador:

1. As Políticas de Agendamento permitem que você configure Predicados para filtragem e Prioridades para pontuação.
2. Os Perfis de Agendamento permitem que você configure Plugins que implementam diferentes estágios de agendamento, incluindo: QueueSort, Filtro, Pontuação, Vinculação, Reserva, Permissão e outros. Você também pode configurar o kube-scheduler para executar diferentes perfis.

(CESPE – TRT - 8ª Região (PA e AP) – 2022) Em Kubernets, o componente que gerencia os pods que foram criados e que está sem nenhum nó atribuído é o

a) kube-scheduler.
b) kube-apiserver.
c) etcd.
d) kube-controller-manager.
e) cloud-controller-manager.

Comentários:

O componente do Kubernetes que gerencia os Pods que foram criados e que ainda não possuem nenhum nó atribuído é o kube-scheduler. O kube-scheduler é responsável por tomar decisões sobre quais nós os Pods devem ser agendados com base em critérios como requisitos de recursos, afinidades, anti-afinidades e restrições. Ele faz parte do processo de agendamento do Kubernetes, garantindo que os Pods sejam alocados nos nós apropriados para melhorar a eficiência, o desempenho e o equilíbrio de carga no cluster. Portanto, a resposta correta é a opção a) kube-scheduler. (Gabarito: Letra A)

kube-controller-manager

Componente do plano de controle que executa processos de controladores. Logicamente, cada controlador é um processo separado, mas para reduzir a complexidade, todos eles são compilados em um único binário e executados em um único processo.

Existem muitos tipos diferentes de controladores. Alguns exemplos são:

- Controlador de nó: Responsável por perceber e responder quando os nós estão inativos.
- Controlador de trabalho (job): Observa os objetos de trabalho que representam tarefas únicas e, em seguida, cria Pods para executar essas tarefas até a conclusão.
- Controlador de EndpointSlice: Popula objetos EndpointSlice (para fornecer um link entre Serviços e Pods).
- Controlador de ServiceAccount: Cria ServiceAccounts padrão para novos namespaces.

cloud-controller-manager

Um componente do plano de controle do Kubernetes que incorpora lógica de controle específica da nuvem. O gerenciador de controladores de nuvem permite que você conecte seu cluster à API do provedor de nuvem e separa os componentes que interagem com a plataforma de nuvem daqueles que interagem apenas com o seu cluster.

O cloud-controller-manager executa apenas controladores específicos do provedor de nuvem. Se você estiver executando o Kubernetes em suas próprias instalações ou em um ambiente de aprendizado dentro do seu próprio PC, o cluster não terá um gerenciador de controladores de nuvem.

Assim como o kube-controller-manager, o cloud-controller-manager combina vários loops de controle logicamente independentes em um único binário que você executa como um único processo. Você pode escalar horizontalmente (executar mais de uma cópia) para melhorar o desempenho ou ajudar a tolerar falhas.

Os seguintes controladores podem ter dependências de provedores de nuvem:

Controlador de nó: Para verificar o provedor de nuvem e determinar se um nó foi excluído na nuvem depois que ele deixa de responder.
Controlador de rota: Para configurar rotas na infraestrutura de nuvem subjacente.
Controlador de serviço: Para criar, atualizar e excluir balanceadores de carga do provedor de nuvem.

Componentes do nó

Os componentes do nó são executados em todos os nós, mantendo os pods em execução e fornecendo o ambiente de tempo de execução do Kubernetes.

kubelet

O kubelet é o principal "agente de nó" que é executado em cada nó. Ele pode registrar o nó no apiserver usando um dos seguintes métodos: o nome do host; uma flag para substituir o nome do host; ou lógica específica para um provedor de nuvem. Ele garante que os contêineres estejam sendo executados em um Pod.

O kubelet opera em termos de um PodSpec. Um PodSpec é um objeto YAML ou JSON que descreve um pod. O kubelet recebe um conjunto de PodSpecs fornecidos por meio de vários mecanismos (principalmente por meio do apiserver) e garante que os containers descritos nesses PodSpecs estejam em execução e saudáveis. O kubelet não gerencia contêineres que não foram criados pelo Kubernetes.

Além de obter um PodSpec do apiserver, existem duas maneiras de fornecer um manifesto de contêiner ao Kubelet.

- Arquivo: Caminho passado como uma flag na linha de comando. Arquivos sob esse caminho serão monitorados periodicamente quanto a atualizações. O período de monitoramento é de 20 segundos por padrão e pode ser configurado por meio de uma flag.

- Ponto de extremidade HTTP: Ponto de extremidade HTTP passado como parâmetro na linha de comando. Esse ponto de extremidade é verificado a cada 20 segundos (também configurável por meio de uma flag).

kube-proxy

O kube-proxy é um proxy de rede do Kubernetes que é executado em cada nó no cluster, implementando parte do conceito de Serviço do Kubernetes. Isso reflete os serviços conforme definidos na API do Kubernetes em cada nó e pode realizar encaminhamento simples de fluxos TCP, UDP e SCTP, ou encaminhamento round robin TCP, UDP e SCTP através de um conjunto de backends. Os IPs e portas dos serviços do cluster são atualmente encontrados por meio de variáveis de ambiente compatíveis com Docker-links, especificando as portas abertas pelo proxy de serviço. Há um complemento opcional que fornece um DNS de cluster para esses IPs de cluster. O usuário deve criar um serviço com a API do servidor de API (apiserver) para configurar o proxy.

O kube-proxy mantém regras de rede nos nós. Essas regras de rede permitem a comunicação de rede para seus Pods a partir de sessões de rede dentro ou fora do seu cluster. O kube-proxy utiliza a camada de filtragem de pacotes do sistema operacional se estiver disponível. Caso contrário, o kube-proxy encaminha o tráfego por si próprio.

Explicando melhor: quando o kube-proxy recebe uma solicitação de tráfego destinada a um Serviço Kubernetes, ele verifica se existem pods associados a esse Serviço. Se houver, o kube-proxy atua como intermediário entre a solicitação de tráfego externo e os pods internos. Ele redireciona o tráfego para o pod apropriado, utilizando as regras de balanceamento de carga e as informações de rede que ele mantém.

O kube-proxy também gerencia as conexões de rede e os endereços IP para garantir que o tráfego seja roteado corretamente e que os pods certos recebam as solicitações. Isso significa que, quando o kube-proxy "encaminha o tráfego por si próprio", ele age como um proxy de rede, interceptando as solicitações e redirecionando-as conforme necessário. Isso permite que os Serviços Kubernetes sejam acessíveis de maneira consistente e eficaz, mesmo quando há alterações na localização ou no número de pods subjacentes.

kube-apiserver

O servidor de API do Kubernetes valida e configura dados para os objetos de API, que incluem pods, serviços, controladores de replicação e outros. O servidor de API oferece serviços para operações REST e fornece a interface frontal para o estado compartilhado do cluster, por meio do qual todos os outros componentes interagem.

kube-controller-manager

O gerenciador de controladores do Kubernetes é um daemon que incorpora os ciclos de controle principais fornecidos com o Kubernetes. Em aplicações de robótica e automação, um ciclo de controle é um loop não terminante que regula o estado do sistema.

No Kubernetes, um controlador é um ciclo de controle que observa o estado compartilhado do cluster por meio do servidor de API (apiserver) e faz alterações na tentativa de mover o estado atual em direção ao estado desejado. Exemplos de controladores que vêm com o Kubernetes hoje são o controlador de replicação, controlador de endpoints, controlador de namespace e controlador de contas de serviço (serviceaccounts).

Pod

Classes de Qualidade de Serviço de Pods

O Kubernetes classifica os Pods que você executa e aloca cada Pod em uma classe específica de qualidade de serviço (QoS). O Kubernetes utiliza essa classificação para influenciar como diferentes pods são tratados. O Kubernetes realiza essa classificação com base nas solicitações de recursos dos Containers nesse Pod, juntamente com a relação dessas solicitações com os limites de recursos. Isso é conhecido como classe de Qualidade de Serviço (QoS). O Kubernetes atribui a cada Pod uma classe de QoS com base nas solicitações e limites de recursos de seus Containers componentes. As classes de QoS são usadas pelo Kubernetes para decidir quais Pods remover de um Node que está sob pressão. As possíveis classes de QoS são Guaranteed, Burstable, and BestEffort (em português Garantida, Expansível e Melhor Esforço, respectivamente). Quando um Node fica sem recursos, o Kubernetes primeiro remove os Pods de Melhor Esforço em execução nesse Node, seguidos pelos Pods Expansíveis e, por fim, pelos Pods Garantidos. Quando essa remoção ocorre devido à pressão de recursos, apenas os Pods que excedem as solicitações de recursos são considerados como candidatos para remoção.

Guaranteed ou Garantido

Pods que são classificados como Garantidos possuem os limites de recursos mais rigorosos e têm menor probabilidade de serem desligados. Eles têm a garantia de que não serão encerrados até que excedam seus limites ou não haja Pods de prioridade inferior que possam ser preemptados do Nó. Eles não podem adquirir recursos além de seus limites especificados. Esses Pods também podem usar CPUs exclusivas usando a política de gerenciamento estático de CPU.

Critérios
Para que um Pod receba uma classe QoS de Garantido:

- Cada Contêiner no Pod deve ter um limite de memória e uma solicitação de memória.
- Para cada Contêiner no Pod, o limite de memória deve ser igual à solicitação de memória.
- Cada Contêiner no Pod deve ter um limite de CPU e uma solicitação de CPU.
- Para cada Contêiner no Pod, o limite de CPU deve ser igual à solicitação de CPU.

Burstable ou Expansível

Pods que são classificados como Expansíveis possuem algumas garantias de recursos mínimos com base na solicitação, mas não exigem um limite específico. Se um limite não for especificado, ele será definido como um limite equivalente à capacidade do Nó, o que permite que os Pods aumentem flexivelmente seus recursos, se estiverem disponíveis. No caso de evacuação de Pod devido à pressão de recursos do Nó, esses Pods são evacuados somente após todos os Pods de Melhor Esforço serem evacuados. Como um Pod Expansível pode incluir um Contêiner que não

possui limites ou solicitações de recursos, um Pod classificado como Expansível pode tentar usar qualquer quantidade de recursos do nó.

Um Pod recebe uma classe QoS de Expansível se:

- O Pod não atende aos critérios para a classe QoS Garantido.
- Pelo menos um Contêiner no Pod possui uma solicitação ou limite de memória ou CPU.

BestEffort ou Melhor Esforço

Pods na classe QoS de Melhor Esforço podem usar recursos do nó que não foram especificamente atribuídos a Pods em outras classes QoS. Por exemplo, se você tem um nó com 16 núcleos de CPU disponíveis para o kubelet e atribui 4 núcleos de CPU a um Pod Garantido, então um Pod na classe QoS de Melhor Esforço pode tentar usar qualquer quantidade dos 12 núcleos de CPU restantes. O kubelet prefere evacuar Pods de Melhor Esforço se o nó estiver sob pressão de recursos.

Um Pod possui uma classe QoS de Melhor Esforço se ele não atender aos critérios para as classes Garantido ou Expansível. Em outras palavras, um Pod é de Melhor Esforço apenas se nenhum dos Contêineres no Pod tiver um limite ou solicitação de memória, e nenhum dos Contêineres no Pod tiver um limite ou solicitação de CPU. Contêineres em um Pod podem solicitar outros recursos (não CPU ou memória) e ainda serem classificados como Melhor Esforço.

Segurança

Controlando o acesso à API do Kubernetes

Como o Kubernetes é totalmente orientado por API, controlar e limitar quem pode acessar o cluster e quais ações eles têm permissão para realizar é a primeira linha de defesa.

Utilize a Segurança da Camada de Transporte (TLS) para todo o tráfego da API

O Kubernetes espera que toda a comunicação da API no cluster seja criptografada por padrão com TLS, e a maioria dos métodos de instalação permitirá que os certificados necessários sejam criados e distribuídos para os componentes do cluster. Note que alguns componentes e métodos de instalação podem habilitar portas locais via HTTP, e os administradores devem se familiarizar com as configurações de cada componente para identificar tráfego potencialmente não seguro.

Autenticação da API

Escolha um mecanismo de autenticação para os servidores de API usarem que corresponda aos padrões comuns de acesso ao instalar um cluster. Por exemplo, clusters pequenos e de único usuário podem desejar usar uma abordagem simples com certificado ou token Bearer estático. Clusters maiores podem desejar integrar um servidor existente de OIDC ou LDAP que permita que os usuários sejam subdivididos em grupos.

Todos os clientes da API devem ser autenticados, mesmo aqueles que fazem parte da infraestrutura, como nós, proxies, o escalonador e plugins de volume. Esses clientes geralmente são contas de serviço ou usam certificados de cliente x509, e são criados automaticamente no início do cluster ou configurados como parte da instalação do cluster.

Autorização da API

Após a autenticação, cada chamada de API também deve passar por uma verificação de autorização. O Kubernetes inclui um componente integrado de Controle de Acesso Baseado em Função (RBAC) que associa um usuário ou grupo recebido a um conjunto de permissões agrupadas em funções. Essas permissões combinam verbos (obter, criar, excluir) com recursos (pods, serviços, nós) e podem ser limitadas ao escopo de um namespace ou abranger todo o cluster. Um conjunto de funções prontas é fornecido, oferecendo uma separação razoável das responsabilidades com base nas ações que um cliente pode querer executar. É recomendado que você utilize os autorizadores Node e RBAC juntos, em combinação com o plugin de admissão NodeRestriction.

Assim como na autenticação, funções simples e amplas podem ser apropriadas para clusters menores, mas à medida que mais usuários interagem com o cluster, pode se tornar necessário separar equipes em namespaces diferentes com funções mais limitadas.

Padrões de Segurança para Pods

O Pod Security Standards define três políticas diferentes para abranger amplamente o espectro de segurança. Essas políticas são cumulativas e variam de altamente permissivas a altamente restritivas.

Perfil	Descrição
Privilegiado	Política não restrita, fornecendo o nível mais amplo possível de permissões. Essa política permite escalonamentos de privilégios conhecidos.
Base	Mínima política restritiva que impede escalonamentos de privilégios conhecidos. Permite a configuração padrão (minimamente especificada) do Pod.
Restrito	Política fortemente restrita, seguindo as melhores práticas atuais de endurecimento do Pod.

Mas como configurar um Contexto de Segurança para um Pod ou Contêiner? Um contexto de segurança define configurações de privilégio e controle de acesso para um Pod ou Contêiner. Um contexto de segurança é como um conjunto de regras que determinam quem pode fazer o quê em um Pod ou Contêiner. Vamos dar uma olhada em algumas das coisas que você pode fazer com esse contexto.

- Controle de Acesso Discrecional: Imagine que o acesso a um objeto, como um arquivo, dependa de uma identificação especial, como um crachá. Nesse caso, o ID do usuário (UID) e o ID do grupo (GID) seriam como esses crachás especiais.

- Security Enhanced Linux (SELinux): Você já pensou em objetos tendo rótulos de segurança? Isso é um pouco como colocar etiquetas em coisas para indicar o quão seguro elas são.

- Execução como privilegiado ou não privilegiado: Agora, se um processo é executado como "privilegiado", é como se estivesse usando um crachá que permite fazer muitas coisas especiais. Mas se é "não privilegiado", esse crachá é mais limitado.

- Linux Capabilities: Às vezes, um processo precisa de alguns poderes extras, mas não queremos dar todos os superpoderes do usuário root. Isso seria como dar um crachá com alguns poderes extras, mas não todos.

- AppArmor: Imagina se cada programa tivesse um perfil próprio que define o que ele pode ou não fazer. Isso é como dar a cada programa um manual de regras.

- Seccomp: Pense em um filtro que só permite que um processo use certos pedidos especiais. É como deixar alguém falar apenas algumas palavras em um telefone.

- allowPrivilegeEscalation: Imagine que um processo queira ser mais poderoso do que seu "pai" (processo pai). Aqui, podemos controlar se isso é permitido ou não. E, quando um contêiner é "privilegiado" ou tem CAP_SYS_ADMIN, é como dar uma autorização especial para aumentar seu poder.

- readOnlyRootFilesystem: Às vezes, queremos ter certeza de que ninguém vai mexer nas coisas importantes do sistema. Então, montamos o sistema de arquivos raiz do contêiner como somente leitura, para que ninguém possa fazer alterações.

Definindo o contexto de segurança para um Pod

Para especificar configurações de segurança para um Pod, inclua o campo securityContext na especificação do Pod. O campo securityContext é um objeto PodSecurityContext. As configurações de segurança que você especifica para um Pod se aplicam a todos os Contêineres no Pod.

Configurando permissão de volume e política de alteração de propriedade para pods

Por padrão, o Kubernetes altera recursivamente a propriedade e as permissões do conteúdo de cada volume para corresponder ao fsGroup especificado no securityContext de um Pod quando esse volume é montado. Para volumes grandes, verificar e alterar a propriedade e as permissões pode levar muito tempo, retardando o início do Pod. Você pode usar o campo fsGroupChangePolicy dentro de um securityContext para controlar a maneira como o Kubernetes verifica e gerencia a propriedade e as permissões de um volume.

Utilizando o kubectl exec para obter um shell em um contêiner em execução

Você precisa ter um cluster Kubernetes e a ferramenta de linha de comando kubectl deve ser configurada para se comunicar com seu cluster. É recomendável executar este tutorial em um cluster com pelo menos dois nós que não estejam atuando como hosts do plano de controle.

Obtendo acesso à linha de comando de um contêiner. Aqui vamos criar um Pod que possui um contêiner. O contêiner executa a imagem nginx. Aqui está o arquivo de configuração para o Pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: shell-demo
spec:
  volumes:
  - name: shared-data
    emptyDir: {}
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - name: shared-data
      mountPath: /usr/share/nginx/html
  hostNetwork: true
  dnsPolicy: Default
```

Crie o Pod:

```
kubectl apply -f https://k8s.io/examples/application/shell-demo.yaml
```

Verifique se o contêiner está em execução:

```
kubectl get pod shell-demo
```

Acesse a linha de comando do contêiner em execução:

```
kubectl exec --stdin --tty shell-demo -- /bin/bash
```

No seu terminal, liste o diretório raiz:

```
ls /
```

No seu terminal, experimente com outros comandos. Você pode executar esses comandos de exemplo dentro do contêiner. Aqui estão alguns exemplos:

```
ls /
cat /proc/mounts
cat /proc/1/maps
apt-get update
apt-get install -y tcpdump
tcpdump
apt-get install -y lsof
lsof
apt-get install -y procps
ps aux
ps aux | grep nginx
```

Executando comandos individuais em um contêiner

Em uma janela de comando comum, não na sua linha de comando, liste as variáveis de ambiente no contêiner em execução:

```
kubectl exec shell-demo env
```

Experimente executar outros comandos. Aqui estão alguns exemplos:

```
kubectl exec shell-demo -- ps aux
kubectl exec shell-demo -- ls /
kubectl exec shell-demo -- cat /proc/1/mounts
```

Abrindo um terminal quando um Pod tem mais de um contêiner

Se um Pod tiver mais de um contêiner, use --container ou -c para especificar um contêiner no comando kubectl exec. Por exemplo, suponha que você tenha um Pod chamado meu-pod, e o Pod tenha dois contêineres chamados main-app e helper-app. O comando a seguir abriria um terminal para o contêiner main-app.

```
kubectl exec -i -t meu-pod --container main-app -- /bin/bash
```

Os comandos básicos do Kubernetes oferecem uma maneira de iniciar e gerenciar cargas de trabalho em seu cluster. O comando "run" é essencial para começar a executar uma ou mais instâncias de uma imagem de contêiner no seu ambiente. Ao usar o comando "expose", você cria uma abordagem de balanceamento de tráfego, permitindo que o tráfego seja distribuído uniformemente entre as instâncias em execução. Isso pode ser estendido para criar um proxy de alta disponibilidade, facilitando o acesso externo aos contêineres do seu cluster.

O comando "create" permite a criação de pods com base em definições específicas, enquanto o comando "get" possibilita a visualização dos recursos disponíveis no cluster. O comando "delete", por sua vez, oferece a flexibilidade de remover recursos de maneira granular, seja por nome, arquivo ou rótulo, garantindo um controle preciso sobre o ambiente do Kubernetes.

Função	Descrição
Create	Criar um pod usando os dados em pod.
get	Exibir um ou vários recursos.
run	Criar e executar uma imagem específica em um pod.
expose	Expor um recurso como um novo serviço Kubernetes.
delete	Excluir recursos por nomes de arquivo, stdin, recursos e nomes, ou por recursos e seletor de rótulo.

Gerenciamento de Aplicativos ("App Management") abrange uma série de comandos fundamentais para administrar suas aplicações em um ambiente Kubernetes. Esses comandos permitem a criação, atualização, exclusão e visualização das cargas de trabalho em seu cluster. Ao utilizar essa gama de ferramentas, você ganha a capacidade de controlar todo o ciclo de vida das suas aplicações, desde a implantação inicial até eventuais ajustes e, se necessário, a remoção eficiente de recursos. Isso proporciona um gerenciamento completo e flexível para garantir que suas aplicações estejam sempre funcionando conforme o desejado no ambiente Kubernetes.

Função	Descrição
apply	Aplica a configuração contida em pod.json a um pod.
annotate	Adiciona ou modifica anotações em um recurso.
autoscale	Define ou ajusta a escala automática para um conjunto de réplicas.
debug	Inicia um pod para fins de depuração.
diff	Exibe as diferenças entre duas versões diferentes de um recurso.
edit	Edita um recurso em tempo real usando o editor padrão.
kustomize	Aplica customizações em um manifesto antes da implantação.
label	Adiciona ou modifica etiquetas em um recurso.
patch	Aplica um patch a um recurso no formato JSON ou YAML.
replace	Substitui um recurso existente por um novo.

rollout	Gerencia o histórico de implantação de um recurso.
scale	Ajusta o número de réplicas de um conjunto de réplicas.
set	Configura recursos de aplicativos.
wait	Aguarda até que uma condição específica seja atingida em um ou vários recursos.

Trabalhar com aplicativos no ecossistema Kubernetes (K8s) é otimizado por um conjunto de comandos. O comando attach permite conectar-se a processos em execução dentro de contêineres, facilitando a análise detalhada. Para inspecionar a autorização, o comando auth oferece uma visão detalhada do controle de acesso. A cópia de arquivos entre o host e contêineres é simplificada pelo comando cp, enquanto o comando describe fornece insights aprofundados sobre recursos específicos. A execução de comandos diretamente em pods é viabilizada pelo exec, e a monitorização de logs é realizada pelo logs. O port-forward facilita a interação direta com aplicativos, e o proxy cria um gateway para comunicação com a API do Kubernetes. Por fim, o top exibe o uso de recursos, auxiliando na análise de desempenho de nós e pods. A utilização conjunta destes comandos otimiza o gerenciamento de aplicativos no K8s.

Função	Descrição
attach	Anexa-se a um processo que já está em execução dentro de um contêiner existente.
auth	Inspeciona a autorização.
cp	Copia arquivos e diretórios de e para contêineres.
describe	Exibe informações detalhadas sobre um ou mais recursos.
exec	Executa um comando em um contêiner.
logs	Exibe os registros de um contêiner em um pod ou recurso especificado.
port-forward	Encaminha portas de um pod para o host.
proxy	Cria um servidor proxy ou gateway de nível de aplicativo entre o localhost e o servidor da API do Kubernetes. Também permite servir conteúdo estático em um caminho HTTP especificado. Todos os dados de entrada são direcionados para uma porta e encaminhados para a porta remota do servidor da API do Kubernetes, exceto para o caminho que corresponde ao conteúdo estático.
top	Exibe informações de uso de recursos para nós e pods.

O gerenciamento de clusters no Kubernetes envolve a habilidade de administrar aspectos do ambiente. Com ferramentas como o kubectl, é possível explorar as versões de API disponíveis por meio do comando api-versions, garantir a segurança e confiabilidade do cluster através da manipulação de recursos de certificados com certificate, e obter informações detalhadas sobre o estado do cluster usando o cluster-info. Além disso, é viável controlar a capacidade de agendamento de nós com cordon e realizar a manutenção planejada ao esvaziar nós usando o drain. Para ajustes de gerenciamento de recursos e distribuição de carga, os comandos taint e uncordon oferecem controle sobre as restrições aplicadas aos nós.

Função	Descrição
api-versions	Exibe as versões de API suportadas no servidor, no formato "grupo/versão".
certificate	Modifica recursos de certificados.
approve	Aprova uma solicitação de assinatura de certificado.
cluster-info	Exibe os endereços do plano de controle e serviços com a etiqueta kubernetes.io/cluster-service=true.
cordon	Marca um nó como não agendável.
drain	Esvazia um nó em preparação para manutenção.
taint	Atualiza as restrições em um ou mais nós.
uncordon	Marca um nó como agendável.

As configurações e uso do kubectl são usados na interação eficaz com clusters Kubernetes. Por meio dessas ferramentas, os administradores e desenvolvedores podem ajustar as configurações de função e acesso, explorar recursos experimentais com o alpha, listar recursos disponíveis usando api-resources, otimizar a experiência do usuário com completions, personalizar a configuração por meio do config, entender melhor o comportamento do Kubernetes através do explain, explorar opções específicas do comando com options, estender a funcionalidade através de plugins e verificar a versão do kubectl e do servidor com o version.

Função	Descrição
alpha	Comandos em estágio alfa.
api-resources	Exibe os recursos de API suportados no servidor.
completion	Gerar scripts de conclusão para shell.
config	Gerenciar configurações do Kubectl.

explain	Exibir informações detalhadas sobre recursos.
options	Exibir opções de linha de comando.
plugin	Gerenciar plugins Kubectl.
version	Exibir a versão do Kubectl.
alpha	Comandos em estágio alfa.

Plataforma como Serviço e Container como Serviço

FaaS

A Plataforma como Serviço, ou PaaS, é uma camada intermediária na pilha de serviços em nuvem que fornece um ambiente de desenvolvimento e implantação para os desenvolvedores criarem, testarem e implantarem aplicativos. Com o PaaS, os desenvolvedores podem se concentrar na criação de código e na funcionalidade do aplicativo, sem se preocupar com a infraestrutura subjacente. Isso acelera o desenvolvimento e permite que as equipes se concentrem mais na inovação do que na manutenção de servidores e sistemas operacionais. O modelo "pay as you use" se estende ao PaaS, permitindo que as empresas ajustem os recursos conforme suas necessidades, o que resulta em maior eficiência e flexibilidade.

FaaS, ou Function as a Service, é um modelo de computação em nuvem que permite aos desenvolvedores executar código sem precisar gerenciar ou provisionar servidores. Com FaaS, os desenvolvedores podem simplesmente enviar suas funções de código para a nuvem e a plataforma FaaS se encarrega de implantá-las e escaloná-las conforme a demanda.

Hospedar um aplicativo de software na internet geralmente requer provisionar e gerenciar um servidor virtual ou físico e gerenciar um sistema operacional e processos de hospedagem de servidor web. Com o FaaS, o hardware físico, o sistema operacional de máquina virtual e o gerenciamento de software do servidor web são todos gerenciados automaticamente pelo provedor de serviço de nuvem. Isso permite que os desenvolvedores se concentrem apenas em funções individuais no código de seu aplicativo.

Funcionamento

Em sistemas FaaS (Function as a Service), as funções são executadas em contêineres que são criados e destruídos dinamicamente, conforme necessário. Isso pode causar problemas de desempenho se houver um grande número de solicitações para uma função.

O balanceamento de carga é uma técnica que pode ser usada para distribuir solicitações entre várias instâncias de função. Isso ajuda a garantir que nenhuma instância de função seja sobrecarregada e que todas as solicitações sejam atendidas de forma eficiente.

Em sistemas FaaS baseados em Kubernetes, o balanceamento de carga pode ser implementado usando instâncias de serviço Kubernetes. As instâncias de serviço Kubernetes são recursos que encapsulam um conjunto de pods.

Quando uma solicitação é feita a um serviço Kubernetes, o serviço encaminha a solicitação para uma das instâncias de função que compõem o serviço. O serviço usa um algoritmo de balanceamento de carga para determinar qual instância de função deve receber a solicitação.

Existem vários algoritmos de balanceamento de carga que podem ser usados por serviços Kubernetes. Alguns dos algoritmos de balanceamento de carga mais comuns incluem:

- Round Robin: O algoritmo Round Robin distribui as solicitações entre as instâncias de função em uma sequência circular.
- Least Connections: O algoritmo Least Connections distribui as solicitações para as instâncias de função com o menor número de conexões.
- Weighted Least Connections: O algoritmo Weighted Least Connections distribui as solicitações para as instâncias de função com o menor número de conexões, ponderando as instâncias de função com base na sua capacidade de processamento.

Monitoramento

O Kubernetes fornece uma variedade de recursos de monitoramento, incluindo:

- Metrics Server: O Metrics Server coleta métricas de recursos de nós e pods.
- Prometheus: O Prometheus é um sistema de monitoramento de código aberto que pode ser usado para coletar e visualizar métricas de pods.
- Grafana: O Grafana é uma ferramenta de visualização de dados de código aberto que pode ser usada para visualizar métricas de Prometheus.

Os sistemas FaaS de código aberto baseados em Kubernetes podem usar esses recursos do Kubernetes para monitorar as instâncias de função. Por exemplo, o Metrics Server pode ser usado para coletar métricas de uso de CPU e memória das instâncias de função. O Prometheus e o Grafana podem ser usados para visualizar essas métricas.

Além dos recursos do Kubernetes, os sistemas FaaS de código aberto também podem usar ferramentas de monitoramento de terceiros, como Prometheus, Grafana e ELK Stack. Essas ferramentas podem ser usadas para coletar e visualizar métricas de funções de qualquer sistema FaaS.

- Prometheus: É um sistema de monitoramento e alerta de código aberto que coleta métricas de várias fontes, armazena essas métricas em um formato eficiente e permite a consulta e visualização desses dados. Ele é amplamente utilizado para monitorar ambientes Kubernetes e sistemas distribuídos.

- Grafana: É uma plataforma de análise e visualização de métricas de código aberto que trabalha bem em conjunto com sistemas como Prometheus. Ela permite criar painéis de controle personalizados para monitorar e analisar métricas em tempo real.

- ELK Stack (Elasticsearch, Logstash, Kibana): Essa é uma combinação de ferramentas usadas para coletar, processar, armazenar e visualizar logs. O Elasticsearch é usado para armazenar e pesquisar logs, o Logstash é usado para processar e encaminhar logs, e o Kibana é usado para criar visualizações e painéis de controle baseados nos dados de log.

Autoescalonamento

O autoescalonamento é uma funcionalidade que permite que o sistema ajuste automaticamente o número de instâncias de função em execução de acordo com a carga de trabalho. A maioria dos sistemas FaaS utiliza recursos de autoescalonamento, como o Horizontal Pod Autoscaling (HPA) integrado do Kubernetes. O HPA monitora métricas de recursos, como CPU e memória, das instâncias de função em execução. Com base nessas métricas, ele pode aumentar ou diminuir o número de instâncias para otimizar o desempenho e a utilização dos recursos. Por exemplo, se a carga de trabalho aumenta, o HPA pode adicionar mais instâncias para lidar com o aumento do tráfego. Se a carga diminui, ele pode reduzir o número de instâncias para economizar recursos.

O HPA é baseado em um objetivo de métricas. O objetivo de métricas é uma métrica que o HPA monitora para determinar se o serviço deve ser escalonado. O HPA pode ser usado para monitorar métricas de recursos de computação, como uso de CPU, memória e armazenamento.

Quando o valor da métrica atinge o objetivo, o HPA pode aumentar ou diminuir o número de pods no serviço. O HPA pode aumentar o número de pods adicionando novos pods ao serviço. O HPA pode diminuir o número de pods removendo pods do serviço.

Além do HPA, os sistemas FaaS também podem usar mecanismos de escalonamento personalizados. Esses mecanismos podem ser usados para escalar os sistemas FaaS com base em métricas diferentes ou com base em regras personalizadas. Por exemplo, um sistema FaaS pode usar um mecanismo de escalonamento personalizado para escalar as funções com base no número de solicitações simultâneas em andamento. Esse mecanismo pode ajudar a garantir que os sistemas FaaS sejam dimensionados de forma adequada para lidar com o tráfego de pico. Vejamos alguns benefícios do uso de mecanismos de escalonamento personalizados em sistemas FaaS:

- Flexibilidade: Os mecanismos de escalonamento personalizados podem ser usados para escalar os sistemas FaaS com base em métricas diferentes ou com base em regras personalizadas.
- Eficiência: Os mecanismos de escalonamento personalizados podem ajudar a escalar os sistemas FaaS de forma mais eficiente.
- Redução de custos: Os mecanismos de escalonamento personalizados podem ajudar a reduzir os custos, escalando os sistemas FaaS apenas quando necessário.

OpenFaaS

Os sistemas FaaS (Function-as-a-Service) de código aberto baseados em Kubernetes compartilham várias características em comum que os tornam eficientes e flexíveis para implantação e gerenciamento de funções. Algumas das características comuns incluem:

1. Portabilidade e Escalabilidade: Esses sistemas permitem que você implante funções em contêineres OCI (Open Container Initiative), o que os torna portáteis e independentes da infraestrutura subjacente. Isso permite que você implante as funções em diferentes ambientes, seja em nuvens públicas, privadas ou híbridas. Além disso, eles oferecem escalabilidade dinâmica, permitindo que as funções se expandam ou se reduzam conforme a demanda.

2. Suporte a Múltiplas Linguagens: Você pode escrever funções em várias linguagens de programação, desde Go, Java, Python e Node.js até outras linguagens como C#, Ruby, PHP e muito mais. Isso dá aos desenvolvedores a liberdade de escolher a linguagem mais adequada para suas necessidades.

3. Automação de Implantação e Escalabilidade: Esses sistemas facilitam a implantação de funções e automatizam o processo de escalabilidade com base na demanda. Eles monitoram a carga de trabalho e podem iniciar, reiniciar, programar e redistribuir automaticamente as funções para garantir que elas atendam às demandas dos usuários.

4. Integração com Eventos: Os sistemas FaaS de código aberto baseados em Kubernetes permitem a invocação de funções em resposta a eventos. Isso significa que as funções podem ser acionadas por eventos de várias fontes, como Apache Kafka, AWS SQS, Postgresql, Cron e MQTT, tornando-os adequados para cargas de trabalho orientadas a eventos.

5. Isolamento e Segurança: Esses sistemas garantem isolamento entre as funções e fornecem medidas de segurança, como políticas de rede e limites de recursos, para garantir que as funções executem de forma segura e não afetem umas às outras.

6. Ecossistema de Ferramentas e Comunidade Ativa: Eles têm um ecossistema rico de ferramentas e recursos, incluindo templates de função, bibliotecas, extensões e uma comunidade ativa que contribui com melhorias e suporte contínuos.

7. Monitoramento e Visibilidade: Geralmente, esses sistemas fornecem soluções para monitorar e visualizar o desempenho das funções. Isso ajuda os desenvolvedores e operadores a entender como suas funções estão se comportando e otimizá-las conforme necessário.

8. Suporte ao Ciclo de Vida: Eles oferecem ferramentas para gerenciar o ciclo de vida das funções, desde a criação e implantação até a monitorização, ajuste de escalabilidade e desativação quando não estão sendo usadas.

9. Compatibilidade com Kubernetes: Esses sistemas aproveitam a robustez e a flexibilidade do Kubernetes para gerenciar clusters de contêineres. Isso permite que você se beneficie das capacidades de orquestração, escalabilidade e gerenciamento de recursos do Kubernetes.

CaaS

Containers como serviço (CaaS) é um serviço em nuvem que ajuda a gerenciar e implantar aplicações usando abstração baseada em container. O CaaS pode ser implantado on-premises ou na nuvem. O provedor oferece o framework ou a plataforma de orquestração na qual os containers serão implantados e gerenciados. É por meio dessa orquestração que as principais funções de TI serão automatizadas.

Uma solução de CaaS é útil, principalmente, para que os desenvolvedores possam criar aplicações em containers mais seguras e escaláveis. Os usuários podem adquirir apenas os recursos que querem (funcionalidades de programação, balanceamento de carga etc.) para economizar e aumentar a eficiência. Os containers criam ambientes consistentes para acelerar o desenvolvimento e a entrega de aplicações nativas em nuvem que possam ser executadas em qualquer ambiente.

O Container como Serviço, ou CaaS, é uma abordagem que oferece aos desenvolvedores a capacidade de empacotar, implantar e executar aplicativos em contêineres, proporcionando isolamento de recursos e garantindo que os aplicativos sejam executados consistentemente em ambientes diferentes.

Com o CaaS, as empresas podem obter vantagens de escalabilidade rápida e gerenciamento simplificado, pois os contêineres podem ser implantados em diversos ambientes sem a necessidade de modificar o código. Isso permite que as equipes de desenvolvimento foquem na criação de aplicativos de maneira mais eficiente e eficaz, enquanto a infraestrutura é gerenciada pelo provedor de serviços em nuvem.

O CaaS também facilita a portabilidade de aplicativos entre diferentes ambientes de nuvem, melhorando a agilidade e flexibilidade das empresas na implantação e gerenciamento de suas aplicações.

(FGV – SEF MG– 2023) Várias plataformas FaaS de código aberto foram desenvolvidas como alternativa às soluções serverless dos provedores de nuvem pública. As plataformas FaaS de

código aberto são quase exclusivamente implementadas sobre o Kubernetes, como por exemplo o OpenFaaS.

Em relação às características que possuem em comum os sistemas FaaS de código aberto baseados em kubernetes, assinale a afirmativa incorreta.

a) As invocações às funções são roteadas por um gateway, definido pelo sistema FaaS ou por uma instância Kubernetes Ingress.
b) O balanceamento de carga das requisições entre instâncias de função é implementado aproveitando as instâncias de serviço Kubernetes.
c) Para o autoescalonamento, a maioria dos sistemas FaaS usa o Horizontal Pod Autoscaling (HPA) integrado do Kubernetes, que implementa o autoescalonamento baseado em recursos de computação das instâncias das funções.
d) Para o autoescalonamento, mecanismos de escalonamento personalizados também podem ser implementados, como escalonamento automático com base no número de solicitações simultâneas em andamento.
e) Para monitorar as instâncias de função, a maioria dos sistemas FaaS de código aberto utiliza suas próprias soluções de captura de métricas e visualização.

Comentários:

FaaS, ou Function as a Service, é um modelo de computação em nuvem que permite aos desenvolvedores executar código sem precisar gerenciar ou provisionar servidores. Com FaaS, os desenvolvedores podem simplesmente enviar suas funções de código para a nuvem e a plataforma FaaS se encarrega de implantá-las e escaloná-las conforme a demanda.

Os sistemas FaaS de código aberto baseados em Kubernetes geralmente usam o Kubernetes para monitorar as instâncias de função. O Kubernetes fornece uma variedade de recursos de monitoramento, incluindo:
Metrics: O Kubernetes coleta métricas sobre o uso de recursos de cada instância de função. Essas métricas podem ser usadas para identificar instâncias de função que estão usando muitos recursos ou que estão com problemas.

Logging: O Kubernetes coleta logs de cada instância de função. Esses logs podem ser usados para diagnosticar problemas com instâncias de função.
Alerting: O Kubernetes pode ser configurado para enviar alertas quando os níveis de uso de recursos ou os logs atingirem certos valores. Esses alertas podem ser usados para notificar os administradores sobre problemas com instâncias de função.
Os sistemas FaaS de código aberto baseados em Kubernetes também podem usar soluções de monitoramento de terceiros, como Prometheus e Grafana. Essas soluções podem fornecer uma visão mais detalhada do uso de recursos e dos logs das instâncias de função.

Portanto, a afirmação de que a maioria dos sistemas FaaS de código aberto utiliza suas próprias soluções de captura de métricas e visualização é incorreta. Na verdade, os sistemas FaaS de código aberto baseados em Kubernetes geralmente usam o Kubernetes para monitorar as instâncias de função. (Gabarito: Letra E)

REFERÊNCIAS

https://kubernetes.io/docs/

QUESTÕES COMENTADAS

1. (CESPE – SERPRO– 2023) Julgue o item subsequente, referentes a ferramentas de integração assíncrona e contêineres.

   O Kubernetes permite agrupar hosts executados em contêineres Linux (LXC) em clusters Kubernetes, os quais podem conter um kubelet, que é um grupo de um ou mais contêineres implantados em um nó, suportando aplicações que realizam, por exemplo, a transmissão de dados em tempo real pelo Apache Kafka.

Comentários:

O Kubernetes é uma plataforma que orquestra contêineres, coordenando sua implantação e gerenciamento. Contudo, algumas informações na afirmação estão equivocadas. Primeiramente, o Kubernetes não se limita a agrupar hosts com contêineres LXC, podendo usar diferentes runtimes. Além disso, um cluster Kubernetes consiste em nós hospedando contêineres orquestrados, não necessariamente LXC. O kubelet, por sua vez, é um agente em cada nó do cluster, responsável pelo gerenciamento de contêineres, não um grupo deles.

O Kubernetes é uma plataforma que possibilita a formação de clusters, nos quais podem ser agrupados os hosts que executam contêineres Linux (LXC). Estes clusters são capazes de abrigar um componente chamado kubelet, que não constitui um grupo de contêineres implantados em um nó, ao contrário do que a afirmação inicial sugere. Na verdade, o kubelet é um agente presente em cada máquina do cluster e é responsável por ler os manifestos dos contêineres, iniciá-los e executá-los. Ele assegura que os contêineres estejam sendo executados em unidades chamadas Pods, que consistem em um ou mais contêineres implantados em um nó. Todos os contêineres de um Pod compartilham recursos como endereço IP, IPC e nome de host, facilitando a administração e a movimentação dos contêineres dentro do cluster.

Gabarito: ERRADO

---

2. (FGV – SEF MG – 2023) Várias plataformas FaaS de código aberto foram desenvolvidas como alternativa às soluções serverless dos provedores de nuvem pública. As plataformas FaaS de código aberto são quase exclusivamente implementadas sobre o Kubernetes, como por exemplo o OpenFaaS.

   Em relação às características que possuem em comum os sistemas FaaS de código aberto baseados em kubernetes, assinale a afirmativa incorreta.

   a) As invocações às funções são roteadas por um gateway, definido pelo sistema FaaS ou por uma instância Kubernetes Ingress.

b) O balanceamento de carga das requisições entre instâncias de função é implementado aproveitando as instâncias de serviço Kubernetes.
c) Para o autoescalonamento, a maioria dos sistemas FaaS usa o Horizontal Pod Autoscaling (HPA) integrado do Kubernetes, que implementa o autoescalonamento baseado em recursos de computação das instâncias das funções.
d) Para o autoescalonamento, mecanismos de escalonamento personalizados também podem ser implementados, como escalonamento automático com base no número de solicitações simultâneas em andamento.
e) Para monitorar as instâncias de função, a maioria dos sistemas FaaS de código aberto utiliza suas próprias soluções de captura de métricas e visualização.

Comentários:

FaaS, ou Function as a Service, é um modelo de computação em nuvem que permite aos desenvolvedores executar código sem precisar gerenciar ou provisionar servidores. Com FaaS, os desenvolvedores podem simplesmente enviar suas funções de código para a nuvem e a plataforma FaaS se encarrega de implantá-las e escaloná-las conforme a demanda.

Os sistemas FaaS de código aberto baseados em Kubernetes geralmente usam o Kubernetes para monitorar as instâncias de função. O Kubernetes fornece uma variedade de recursos de monitoramento, incluindo:

- Metrics: O Kubernetes coleta métricas sobre o uso de recursos de cada instância de função. Essas métricas podem ser usadas para identificar instâncias de função que estão usando muitos recursos ou que estão com problemas.
- Logging: O Kubernetes coleta logs de cada instância de função. Esses logs podem ser usados para diagnosticar problemas com instâncias de função.
- Alerting: O Kubernetes pode ser configurado para enviar alertas quando os níveis de uso de recursos ou os logs atingirem certos valores. Esses alertas podem ser usados para notificar os administradores sobre problemas com instâncias de função.

Os sistemas FaaS de código aberto baseados em Kubernetes também podem usar soluções de monitoramento de terceiros, como Prometheus e Grafana. Essas soluções podem fornecer uma visão mais detalhada do uso de recursos e dos logs das instâncias de função.

Portanto, a afirmação de que a maioria dos sistemas FaaS de código aberto utiliza suas próprias soluções de captura de métricas e visualização é incorreta. Na verdade, os sistemas FaaS de código aberto baseados em Kubernetes geralmente usam o Kubernetes para monitorar as instâncias de função.

Gabarito: Letra E

---

3. (FGV – SEF MG– 2023) Kubernetes é um sistema de orquestração de contêineres opensource que automatiza a implantação, o dimensionamento e a gestão de aplicações em contêineres.

Em relação ao conceito de Kubernetes, assinale a afirmativa incorreta.

a) Kube-proxy é um proxy de rede executado e mantém regras de rede em cada máquina do cluster
b) Kubernetes utilizam controladores que rastreiam pelo menos um tipo de recurso Kubernetes.
c) Kubernetes utiliza contexts como mecanismo para isolar grupos de recursos dentro de um único cluster.
d) Kubelet é um agente que é executado em cada máquina do cluster; ele garante que os contêineres estejam sendo executados em um Pod.
e) Os objetos do Kubernetes são entidades persistentes no Kubernetes e utilizam estas entidades para representar o estado do cluster.

Comentários:

Pessoal, vamos analisar cada item.

A) Kube-proxy é um proxy de rede executado e mantém regras de rede em cada máquina do cluster: Isso está correto. O Kube-proxy é responsável por gerenciar as regras de rede no nível do serviço e proporcionar a comunicação entre os Pods.

B) Kubernetes utilizam controladores que rastreiam pelo menos um tipo de recurso Kubernetes: Isso está correto. Os controladores do Kubernetes são componentes que monitoram o estado do cluster e trabalham para garantir que o estado desejado seja mantido. Eles operam com base em um tipo de recurso específico, como Deployments, StatefulSets, DaemonSets, etc.

C) Kubernetes utiliza contexts como mecanismo para isolar grupos de recursos dentro de um único cluster: Isso está incorreto. A resposta correta é que os "namespaces" são usados para isolar grupos de recursos dentro de um único cluster. Os "contexts", por outro lado, são usados para gerenciar diferentes configurações de acesso a diferentes clusters Kubernetes.

D) Kubelet é um agente que é executado em cada máquina do cluster; ele garante que os contêineres estejam sendo executados em um Pod: Isso está correto. O Kubelet é responsável por garantir que os contêineres em um Pod estejam em execução nas máquinas do cluster.

E) Os objetos do Kubernetes são entidades persistentes no Kubernetes e utilizam estas entidades para representar o estado do cluster: Isso está correto. Os objetos do Kubernetes são recursos persistentes que representam o estado do cluster, como Pods, Services, ConfigMaps, Deployments, etc.

Assim, a alternativa incorreta é a C, já que o mecanismo para isolar grupos de recursos dentro de um único cluster são os "namespaces", não os "contexts".

Gabarito: Letra C

---

4. (UFU – MG – UFU – MG – 2023) Kubernetes é um plataforma de código aberto, portável e extensiva, utilizada para automatizar a implantação, o dimensionamento e o gerenciamento de cargas de trabalho e serviços distribuídos em contêineres. O cluster Kubernetes consiste em um conjunto de servidores de processamento que executam aplicações conteinerizadas. Esses servidores hospedam Pods, que são componentes de uma aplicação. Considerando as características e a operacionalização do Kubernetes, analise as asserções abaixo.

I. No Kubernetes, as configurações de contexto de segurança especificadas para um POD não se aplicam a todos os Contêineres no POD, mas somente aos referidos no arquivo de configuração do POD.

II. A linha de comando "kubectl exec -it POD_NAME – sh" é utilizada para inicializar um agrupamento de contêineres definido em um POD denominado POD_NAME.

III. O Kubernetes fornece diferentes níveis de Qualidade de Serviço aos PODs por meio da classificação de cada POD em uma classe QoS específica. Três classes denominadas como "Burstable", "Guaranteed" e "BestEffort são definidas com base nas solicitações de recursos e nos limites de recursos estabelecidos para os Contêineres do POD.

IV. O Kubernetes utiliza as classes QoS para tomar decisões sobre quais PODs despejar quando não houver recursos suficientes disponíveis em um nó. Em situações de falta de recursos, o Kubernetes interromperá a execução de PODs de acordo com a classificação atribuída, sendo primeiro removidos aqueles classificados como "BestEffort", seguido pelos "Guaranteed", e por fim, os "Burstable".

Marque a alternativa que classifica corretamente as asserções como verdadeiras (V) ou falsas (F)

a) I - V; II - V; III - F; IV - F.
b) I - F; II - F; III - V; IV - F.
c) I - V; II - F; III - F; IV - V.
d) I - F; II - V; III - V; IV - V.

Comentários:

Vamos analisar cada item. I. No Kubernetes, as configurações de contexto de segurança especificadas para um POD não se aplicam a todos os Contêineres no POD, mas somente aos

referidos no arquivo de configuração do POD. Esta afirmação é falsa. As configurações de contexto de segurança (como as configurações de permissões) se aplicam a todos os contêineres dentro de um POD. Não é possível especificar configurações de segurança individualizadas para contêineres dentro do mesmo POD.

II. A linha de comando "kubectl exec -it POD_NAME – sh" é utilizada para inicializar um agrupamento de contêineres definido em um POD denominado POD_NAME. Esta afirmação é falsa. A linha de comando "kubectl exec" é usada para executar um comando dentro de um contêiner que já está em execução em um POD. Não é usada para inicializar um agrupamento de contêineres em um POD.

III. O Kubernetes fornece diferentes níveis de Qualidade de Serviço aos PODs por meio da classificação de cada POD em uma classe QoS específica. Três classes denominadas como "Burstable", "Guaranteed" e "BestEffort" são definidas com base nas solicitações de recursos e nos limites de recursos estabelecidos para os Contêineres do POD. Esta afirmação é verdadeira. O Kubernetes classifica os PODs em diferentes níveis de Qualidade de Serviço (QoS) com base nas solicitações e limites de recursos definidos para os contêineres. As três classes QoS são "Burstable", "Guaranteed" e "BestEffort".

IV. O Kubernetes utiliza as classes QoS para tomar decisões sobre quais PODs despejar quando não houver recursos suficientes disponíveis em um nó. Em situações de falta de recursos, o Kubernetes interromperá a execução de PODs de acordo com a classificação atribuída, sendo primeiro removidos aqueles classificados como "BestEffort", seguido pelos "Guaranteed", e por fim, os "Burstable". Esta afirmação é falsa. Em situações de falta de recursos, o Kubernetes não interrompe a execução de PODs com base nas classes QoS. Ele utiliza um processo chamado "culling" para liberar recursos, onde os PODs menos prioritários (normalmente "BestEffort" e "Burstable") podem ser encerrados primeiro, mas a ordem não é necessariamente como descrito na afirmação.

Gabarito: Letra B

---

5. (CESPE – SEFIN de Fortaleza - CE – 2023) Tendo em vista que, no atual cenário de desenvolvimento de aplicações web, é essencial considerar princípios, como consistência e escalabilidade, e práticas, como automação do processo de implantação e integração do código-fonte, julgue o item subsequente.

   As ferramentas DevOps incluem o Kubernetes, uma plataforma de orquestração de contêineres que permite gerenciar e escalonar aplicativos em contêineres em diferentes ambientes e pode ser integrada a outras ferramentas DevOps, como Docker e Jenkins.

Comentários:

O Kubernetes é uma ferramenta essencial dentro da metodologia DevOps, pois oferece recursos avançados de automação e gerenciamento de contêineres. Ele permite aos desenvolvedores implantar, gerenciar e escalar aplicativos em contêineres de maneira eficiente, garantindo que os ambientes de desenvolvimento, teste e produção estejam alinhados. Isso promove a agilidade, a padronização e a consistência ao longo do ciclo de vida do aplicativo.

Gabarito: Correto

---

6. (INSTITUTO AOCP – IF-MA – 2023) No Kubernetes, qual é a abordagem correta para implementar comunicação segura entre os componentes do cluster e garantir autenticação e autorização adequadas?

a) Desabilitar o controle de acesso baseado em função (RBAC) e permitir o acesso a todos os usuários e serviços.
b) Utilizar certificados autoassinados para todos os componentes do cluster e ignorar a validação de certificados.
c) Implementar políticas de segurança usando o etcd como armazenamento centralizado de credenciais e chaves de criptografia.
d) Configurar o Kubernetes para utilizar criptografia TLS, certificados válidos e controle de acesso baseado em função (RBAC).
e) Utilizar somente redes não criptografadas entre os componentes do cluster para melhorar o desempenho.

Comentários:

Pessoal, a abordagem correta para implementar comunicação segura entre os componentes de um cluster Kubernetes, garantindo autenticação e autorização adequadas, é a opção (d) - "Configurar o Kubernetes para utilizar criptografia TLS, certificados válidos e controle de acesso baseado em função (RBAC)".

A criptografia TLS é uma tecnologia que fornece privacidade e integridade para a comunicação entre dois computadores. Ela funciona criptografando os dados antes de serem transmitidos e descriptografando-os após o recebimento. Isso ajuda a proteger os dados contra interceptação e modificação não autorizada.

O uso de certificados válidos é essencial para a segurança da criptografia TLS. Os certificados fornecem identificação e validação para os computadores que se comunicam. Eles também ajudam a garantir que os dados não sejam interceptados e modificados por um usuário não autorizado.

O controle de acesso baseado em função (RBAC) é um mecanismo que permite aos administradores do Kubernetes controlar quem pode acessar quais recursos. Ele funciona

associando funções a usuários e grupos. As funções definem as permissões que um usuário ou grupo tem.

As opções (a), (b) e (e) não atendem aos requisitos de segurança. A opção (a) desabilita o RBAC, o que significa que qualquer usuário ou serviço terá acesso a todos os recursos do cluster. Isso é inseguro, pois permite que usuários não autorizados acessem dados e recursos confidenciais.

A opção (b) utiliza certificados autoassinados, que são certificados que não são emitidos por uma autoridade de certificação confiável. Os certificados autoassinados podem ser facilmente falsificados, o que torna a criptografia TLS menos segura. A opção (e) utiliza redes não criptografadas, o que torna possível a interceptação e modificação não autorizada de dados.

Gabarito: Letra D

---

7. (FGV – MPE-GO – 2022) Uma equipe de especialistas em infraestrutura está implantando o orquestrador Kubernetes em um servidor do MP-GO que possui vários containers Docker. Foi decidido que alguns desses containers serão agrupados na unidade básica de operações do Kubernetes.

   Para isso, a equipe deve definir um novo

   a) statefulSet.
   b) replicaSet.
   c) service.
   d) ingress.
   e) pod.

Comentários:

A resposta correta é a letra E. Um pod é a unidade básica de operações do Kubernetes. Ele é um grupo de um ou mais containers Docker que são executados juntos em um ambiente compartilhado.

Os outros itens não são adequados para este cenário. Um statefulSet é um tipo de replicaSet que garante a disponibilidade de um número especificado de pods, mesmo que alguns deles falhem. Um replicaSet é um grupo de pods que são executados em paralelo para fornecer alta disponibilidade para um aplicativo. Um service é um recurso do Kubernetes que fornece um endereço IP e um nome DNS para um conjunto de pods. Um ingress é um recurso do Kubernetes que roteia o tráfego para um conjunto de pods.

Para agrupar alguns dos containers Docker em uma unidade básica de operações do Kubernetes, a equipe de especialistas deve definir um pod. O pod irá agrupar os containers Docker juntos e fornecer um ambiente compartilhado para eles.

Gabarito: Letra E

---

8. (FEPESE – FAPESC – 2022) A utilização de Kubernetes no gerenciamento de aplicações está sendo amplamente difundida.

Com relação ao Kubernetes, é correto afirmar:

a) O Kubernetes não realiza o escalonamento e a recuperação no caso de falhas.
b) Os fluxos de trabalho de integração contínua, entrega e implantação (CI/CD) são determinados pelo Kubernetes.
c) Fornece serviços em nível de aplicação, tais como middlewares, bancos de dados e caches.
d) Limita o tipo de aplicações executadas a aplicações statefull.
e) Se o tráfego para um contêiner gerenciado pelo Kubernetes for alto, ele pode balancear a carga e distribuir o tráfego de rede para que a implantação seja

Comentários:

A resposta correta é a letra (e), pois o Kubernetes é capaz de balancear a carga e distribuir o tráfego de rede para que a implantação seja mais eficiente. O Kubernetes é um sistema de orquestração de contêineres que automatiza o gerenciamento de aplicativos em contêineres. Ele fornece uma variedade de recursos para ajudar a garantir que os aplicativos sejam executados de forma confiável, incluindo escalabilidade, recuperação de falhas e balanceamento de carga.

O Kubernetes pode balancear a carga entre vários pods de um serviço. Um pod é um grupo de contêineres que são executados juntos em um nó. O Kubernetes pode distribuir o tráfego entre os pods de um serviço com base em vários critérios, como a carga de CPU ou memória de cada pod. O Kubernetes também pode balancear a carga entre vários serviços. Um serviço é um grupo de pods que fornecem o mesmo conjunto de recursos. O Kubernetes pode distribuir o tráfego entre os serviços com base em vários critérios, como o número de solicitações recebidas por cada serviço.

O balanceamento de carga é uma técnica importante para garantir que os aplicativos sejam executados de forma eficiente e escalável. Ele ajuda a garantir que as solicitações sejam distribuídas uniformemente entre os pods e serviços, evitando que um pod ou serviço seja sobrecarregado.

Vejamos por que as outras opções são incorretas:

(a) O Kubernetes realiza o escalonamento e a recuperação no caso de falhas.
(b) Os fluxos de trabalho de CI/CD não são determinados pelo Kubernetes. Eles são definidos pelo desenvolvedor ou equipe de operações.
(c) O Kubernetes não fornece serviços em nível de aplicação, tais como middlewares, bancos de dados e caches. Esses serviços são fornecidos por outros componentes, como o Docker Swarm ou o Amazon Elastic Container Service (ECS).
(d) O Kubernetes não limita o tipo de aplicações executadas a aplicações statefull. Ele pode ser usado para executar qualquer tipo de aplicativo, incluindo aplicações stateless e stateful.

Gabarito: Letra E

---

9. (FGV – MPE-GO – 2022) Uma equipe de especialistas em infraestrutura está implantando o orquestrador Kubernetes em um servidor do MP-GO que possui vários containers Docker. Foi decidido que alguns desses containers serão agrupados na unidade básica de operações do Kubernetes.

Para isso, a equipe deve definir um novo

a) statefulSet.
b) replicaSet.
c) service.
d) ingress.
e) pod.

Comentários:

A resposta correta é a letra (e), pois o pod é a unidade básica de operações do Kubernetes. Um pod é um grupo de um ou mais contêineres Docker que são executados juntos em um nó do Kubernetes. Os pods são projetados para serem leves e portáteis, e podem ser facilmente implantados, escalados e gerenciados pelo Kubernetes.

Para agrupar os containers Docker no MP-GO, a equipe de infraestrutura deve criar um novo pod. O pod deve especificar os containers que devem ser agrupados, bem como as configurações de rede e armazenamento para os containers.

Os demais itens estão incorretos, vejamos:

(a) O statefulSet é um recurso do Kubernetes que permite que você implante e gerencie um conjunto de pods que compartilham um estado.
(b) O replicaSet é um recurso do Kubernetes que permite que você implante e gerencie um conjunto de pods com o mesmo conjunto de configurações.
(c) O service é um recurso do Kubernetes que fornece um ponto de acesso para um grupo de pods.

(d) O ingress é um recurso do Kubernetes que permite que você redirecione o tráfego para diferentes serviços em um cluster Kubernetes.

Gabarito: Letra E

---

10. (FEPESE – Polícia Científica - SC – 2022) Assinale a alternativa que define corretamente o Kubernetes.

a) É um tipo de malware que abre uma ou mais portas para acesso remoto de modo a executar código ou aplicações maliciosas.
b) É uma arquitetura de nuvem pública que possibilita a virtualização de infraestrutura e o seu provimento como serviço (Sas).
c) É uma técnica de Phishing na qual a vítima é induzida a prover dados confidenciais como senhas utilizando-se de engenharia social.
d) É um Desktop Virtual que provê um ambiente seguro e monitorado como estação de trabalho para utilização em home-office.
e) É um orquestrador que possibilita automatizar a implantação e escala de aplicações em containers.

Comentários:

A resposta correta é a letra (e). Kubernetes é um orquestrador de contêineres que automatiza a implantação e escala de aplicações em containers. Ele fornece uma variedade de recursos para ajudar a garantir que os aplicativos sejam executados de forma confiável, incluindo escalabilidade, recuperação de falhas e balanceamento de carga.

As outras opções são incorretas:

(a) Um malware que abre uma ou mais portas para acesso remoto é chamado de backdoor.
(b) Uma arquitetura de nuvem pública que possibilita a virtualização de infraestrutura e o seu provimento como serviço (Sas) é chamada de Infrastructure as a Service (IaaS).
(c) Uma técnica de Phishing na qual a vítima é induzida a prover dados confidenciais como senhas utilizando-se de engenharia social é chamada de Phishing de engenharia social.
(d) Um Desktop Virtual que provê um ambiente seguro e monitorado como estação de trabalho para utilização em home-office é chamado de Virtual Desktop Infrastructure (VDI).

Gabarito: Letra E

---

11. (FGV – TJ-DFT – 2022) A analista Ana precisa implantar o pod LogPod no cluster de Kubernetes KCluster do TJDFT de forma que todos os nós elegíveis do KCluster executem uma cópia do LogPod.

Para que o KCluster apresente uma cópia do LogPod em cada nó elegível, de forma automática, Ana deve implantar o LogPod utilizando o recurso do Kubernetes:

a) ReplicationController;
b) ReplicaSet;
c) StatefulSet;
d) DaemonSet;
e) EndpointSlice.

Comentários:

A resposta correta é a letra (d), DaemonSet. Um DaemonSet é um recurso do Kubernetes que garante que um pod seja executado em todos os nós elegíveis de um cluster. O DaemonSet é usado para executar tarefas de manutenção em todos os nós, como monitoramento ou depuração.

No caso da analista Ana, ela deseja que o pod LogPod seja executado em todos os nós elegíveis do KCluster. Para isso, ela deve implantar o LogPod usando um DaemonSet. O DaemonSet garantirá que uma cópia do LogPod seja executada em todos os nós elegíveis do cluster.

As outras opções são incorretas:

Um ReplicationController garante que um número específico de pods seja executado em um cluster.
Um ReplicaSet garante que um número específico de pods seja executado em um cluster, mas ele não garante que os pods sejam executados em todos os nós elegíveis.
Um StatefulSet garante que pods sejam executados em sequência em um cluster, mas ele não garante que os pods sejam executados em todos os nós elegíveis.
Um EndpointSlice é um recurso do Kubernetes que fornece um ponto de acesso para um conjunto de pods.

Gabarito: Letra D

---

12. (UFMT – POLITEC-MT – 2022) O Kubernetes é utilizado para automatizar o gerenciamento de aplicativos em contêiner. A respeito do Kubernetes, marque V para as afirmativas verdadeiras e F para as falsas.

( ) Utiliza pod para criar um ambiente de teste local.
( ) Usa volumes como máquinas de processamento.
( ) Tem kubectl como interface de linha de comando.
( ) Trata-se de produto comercial, de código fechado.

Assinale a sequência correta.

a) V, F, V, F
b) F, V, V, V
c) V, V, F, V
d) V, F, F, F
e) F, F, V, F

Comentários:

Pessoal, vamos analisar cada alternativa.

(FALSO) Utiliza pod para criar um ambiente de teste local. Um pod é uma unidade básica de processamento no Kubernetes. Ele é composto por um ou mais contêineres que são executados juntos em um nó. Os pods são projetados para serem leves e portáteis, e podem ser facilmente implantados, escalados e gerenciados pelo Kubernetes. Para criar um ambiente de teste local, o Kubernetes pode ser usado para implantar um ou mais pods em um único nó. No entanto, este não é o uso típico do Kubernetes. O Kubernetes é mais frequentemente usado para criar ambientes de produção que podem ser executados em vários nós.

(FALSO) Usa volumes como máquinas de processamento. Um volume é um recurso do Kubernetes que fornece armazenamento para um pod. O armazenamento pode ser fornecido por um disco local, um armazenamento de rede ou um armazenamento em nuvem. Os volumes são usados para armazenar dados que são compartilhados entre os contêineres de um pod. Por exemplo, um volume pode ser usado para armazenar dados de log, dados de banco de dados ou dados de cache. Os volumes não são usados como máquinas de processamento. As máquinas de processamento são responsáveis por executar os contêineres.

(VERDADE) Tem kubectl como interface de linha de comando. O kubectl é uma ferramenta de linha de comando que é usada para gerenciar o Kubernetes. O kubectl pode ser usado para criar, modificar e excluir pods, serviços, nós e outros recursos do Kubernetes.

(FALSO) Trata-se de produto comercial, de código fechado. O Kubernetes é um projeto de código aberto. Isso significa que o código-fonte do Kubernetes está disponível publicamente para qualquer pessoa usar ou modificar. O Kubernetes é mantido pela Cloud Native Computing Foundation (CNCF). A CNCF é uma organização sem fins lucrativos que trabalha para promover o desenvolvimento de tecnologias de computação em nuvem. O Kubernetes não é um produto comercial. Não há uma empresa que venda o Kubernetes como um produto. O Kubernetes é gratuito para usar e modificar.

Gabarito: Letra E

13. (FGV – SENADO– 2022) Uma instituição acadêmica está desenvolvendo um ambiente de laboratório virtual em kubernetes, com alta disponibilidade. O sistema provisiona automaticamente um novo pod quando um aluno se autentica.
Um dos pré-requisitos do projeto é isolar os pods dos alunos em um pool de nodes exclusivo em alta disponibilidade.
Assinale a opção que indica a funcionalidade do kubernetes adequada a esse pré-requisito.

a) Ingress
b) Node affinity
c) Limit Ranges
d) Pod Overhead
e) Node-pressure Eviction

Comentários:

A resposta correta é a letra (b), node affinity. Node affinity é uma funcionalidade do Kubernetes que permite que você especifique restrições sobre quais nós um pod pode ser implantado. No caso da instituição acadêmica, o pré-requisito é isolar os pods dos alunos em um pool de nodes exclusivo em alta disponibilidade. Para isso, a instituição pode usar node affinity para especificar que os pods dos alunos só podem ser implantados em nodes com um determinado rótulo. Por exemplo, a instituição pode criar um rótulo chamado "lab-node" e atribuí-lo a todos os nodes do pool de nodes exclusivo. Em seguida, ela pode especificar no pod spec que o pod só pode ser implantado em nodes com o rótulo "lab-node". Isso garantirá que os pods dos alunos sejam sempre implantados em nodes exclusivos, o que ajudará a garantir a alta disponibilidade e a segurança dos dados.

Gabarito: Letra B

---

14. (FGV – TRT - 13ª Região (PB) – 2022) A infraestrutura recomendada para instalar o Rancher em um cluster Kubernetes K3 de alta disponibilidade é

a) dois nós Linux, normalmente máquina virtual, e um banco de dados externo do tipo NoSQL.
b) um cluster com Linux com menos dois nós ou quatro nós se Windows Server normalmente máquinas virtuais; com pelos menos 64 GB de memória RAM no provedor de infraestrutura e um banco de dados externo do tipo NoSQL.
c) um cluster com Linux de dois nós normalmente máquinas virtuais; com pelos menos 48 GB de memória RAM no provedor de infraestrutura.
d) dois nós Linux, normalmente máquinas virtuais; um banco de dados externo do tipo relacional, um balanceador de carga e um registro DNS.
e)

quatro nós Windows Server, normalmente máquinas virtuais; um banco de dados externo do tipo relacional, um balanceador de carga entre os quatro nós e dois registros DNS como redundância.

Comentários:

De acordo com a documentação oficial do Rancher, a infraestrutura recomendada para instalar o Rancher em um cluster Kubernetes K3 de alta disponibilidade é a seguinte:

Dois nós Linux, normalmente máquinas virtuais;
Um banco de dados externo do tipo relacional, como o MySQL ou o PostgreSQL;
Um balanceador de carga;
Um registro DNS.

O banco de dados externo é necessário para armazenar os dados do Rancher, como usuários, grupos, clusters e aplicativos. O balanceador de carga é necessário para distribuir o tráfego de entrada para os nós do Rancher. O registro DNS é necessário para resolver os nomes de domínio dos nós do Rancher.

Gabarito: Letra D

---

15. **(FCC – TRT - 19ª Região (AL)– 2022)** Nos projetos da Plataforma Digital do Poder Judiciário PDPJ-Br,

a) os ambientes de deployment estão instalados em clusters Kubernetes, gerenciados por meio da ferramenta open source de orquestração de containers Rancher.
b) a aplicação deve possuir os arquivos yanl, para que o deploy ocorra de forma automática por meio de pipelines de CI/CD (Continuous Delivery e Continuous Integration) executados pelo GitLab do STF (Supremo Tribunal Federal).
c) a aplicação deve possuir um arquivo Rancherfile, especificando o modo pelo qual ela deve ser empacotada.
d) deve haver o arquivo .gitlab-ci.xml definindo a pipeline de deploy automático.
e) deve haver, na pasta kubernets, o arquivo de configuração do ingress da aplicação, que é obrigatório.

Comentários:

Pessoal, vamos analisar cada item:

a) Correto: Os ambientes de deployment estão instalados em clusters Kubernetes, gerenciados por meio da ferramenta open source de orquestração de containers Rancher. Isso indica que os

projetos da Plataforma Digital do Poder Judiciário utilizam clusters Kubernetes e utilizam o Rancher para gerenciar esses clusters.

As outras opções estão incorretas:

b) Errado: Embora os arquivos YAML sejam frequentemente usados para definir configurações de recursos no Kubernetes, não é uma exigência absoluta. E o fato de que a aplicação deve possuir arquivos YAML não foi especificamente mencionado.

c) Errado: A existência de um arquivo "Rancherfile" não é uma prática comum em projetos Kubernetes. Geralmente, a configuração de aplicativos Kubernetes é feita por meio de arquivos YAML.

d) Errado: A nomenclatura correta para o arquivo de configuração da pipeline no GitLab é ".gitlab-ci.yml" (com extensão .yml), não ".gitlab-ci.xml".

e) Errado: A pasta padrão para arquivos de configuração relacionados ao Kubernetes é "kubernetes" (com "k" minúsculo), não "kubernets". Além disso, embora a configuração de Ingress seja comum para expor serviços, a obrigatoriedade de um arquivo de configuração de Ingress na pasta não foi mencionada.

Gabarito: Letra A

---

16. (FCC – TRT - 22ª Região (PI) – 2022) No Kubernetes, os contêineres gerenciados pelo kubelet podem usar a estrutura de hook do ciclo de vida do contêiner para executar código acionado por eventos durante o gerenciamento de seu ciclo de vida. Há dois tipos de hooks:

I. Este hook é executado imediatamente após um contêiner ser criado. Mas não há garantia de que o hook será executado antes do ENTRYPOINT do contêiner. Nenhum parâmetro é passado para o handler.

II. Esse hook é chamado imediatamente antes de um contêiner ser terminated devido a uma solicitação de API ou um gerenciamento de evento como liveness/startup probe failure, preemption, resource contention e outros. Uma chamada a este hook falha se o contêiner já está em um estado terminated ou completed e o hook deve ser concluído antes que o sinal TERM seja enviado para parar o contêiner. Nenhum parâmetro é passado para o handler.

Os hooks I e II são, correta e respectivamente,
a) PostStart e PreStop.
b) HookStart e HookStop.
c) PodStart e PodStop.
d) KubeStart e KubeStop.

e) KubeletStart e KubeletStop.

Comentários:

Pessoal, nosso gabarito é a Letra A. Vamos analisar os itens.
I. Este hook é executado imediatamente após um contêiner ser criado. Mas não há garantia de que o hook será executado antes do ENTRYPOINT do contêiner. Nenhum parâmetro é passado para o handler. -> Isso corresponde ao hook PostStart.

II. Esse hook é chamado imediatamente antes de um contêiner ser terminado devido a uma solicitação de API ou um gerenciamento de evento como liveness/startup probe failure, preemption, resource contention e outros. Uma chamada a este hook falha se o contêiner já está em um estado terminated ou completed e o hook deve ser concluído antes que o sinal TERM seja enviado para parar o contêiner. Nenhum parâmetro é passado para o handler. -> Isso corresponde ao hook PreStop.

Os hooks I e II permitem que os desenvolvedores executem ações específicas quando um contêiner é criado ou antes que ele seja terminado no Kubernetes. O hook PostStart é útil para realizar tarefas de inicialização após a criação de um contêiner, enquanto o hook PreStop permite que o contêiner execute ações finais antes de ser encerrado. Estes hooks são uma parte importante da gestão do ciclo de vida de um contêiner em um ambiente Kubernetes.

Gabarito: Letra A

---

17. (FCC – TRT - 17ª Região (ES) – 2022) Considere as seguintes definições:

I. Gerencia os containers em execução e, por isso, ele também é chamado de Orquestrador de Containers. Através dele se pode definir o estado de um sistema completo, por exemplo, baseado em Microservices, seguindo boas práticas de infraestrutura como código, permitindo balanceamento de carga, alta disponibilidade, atualizações em lote e rollbacks.
II. Refere-se a um grupo de servidores distribuídos geograficamente que trabalham em conjunto para oferecer uma rápida entrega de conteúdo de internet. Permite uma transferência rápida dos ativos necessários para carregar o conteúdo da internet, incluindo páginas HTML, arquivos em JavaScript, folhas de estilo, imagens e vídeos.
III. Projeto opensource escrito em Go, que torna a criação e o gerenciamento de containers Linux mais fácil. O container é construído usando namepspaces, cgroups, chroot entre outras funcionalidades do kernel para criar uma área isolada para a aplicação. É executado como um processo isolado no sistema operacional hospedeiro, compartilhando o kernel com outros recipientes.

As definições I, II e III correspondem, correta e respectivamente, a

a) Kubernetes, Docker e Content Delivery Network (CDN).

b) Content Delivery Network (CDN), Kubernetes e Docker.
c) Content Delivery Network (CDN), Docker e Kubernetes.
d) Docker, Content Delivery Network (CDN) e Kubernetes.
e) Kubernetes, Content Delivery Network (CDN) e Docker.

Comentários:

I. Gerencia os containers em execução e, por isso, ele também é chamado de Orquestrador de Containers. Através dele se pode definir o estado de um sistema completo, por exemplo, baseado em Microservices, seguindo boas práticas de infraestrutura como código, permitindo balanceamento de carga, alta disponibilidade, atualizações em lote e rollbacks. -> Isso corresponde ao Kubernetes, que é um orquestrador de contêineres amplamente utilizado para automatizar a implantação, escalonamento e operações de aplicativos em contêineres.

II. Refere-se a um grupo de servidores distribuídos geograficamente que trabalham em conjunto para oferecer uma rápida entrega de conteúdo de internet. Permite uma transferência rápida dos ativos necessários para carregar o conteúdo da internet, incluindo páginas HTML, arquivos em JavaScript, folhas de estilo, imagens e vídeos. -> Isso corresponde a uma Content Delivery Network (CDN), que é uma rede de servidores otimizados para entregar conteúdo de maneira rápida e eficiente aos usuários, geralmente distribuídos geograficamente para melhorar a experiência do usuário.

III. Projeto opensource escrito em Go, que torna a criação e o gerenciamento de containers Linux mais fácil. O container é construído usando namespaces, cgroups, chroot entre outras funcionalidades do kernel para criar uma área isolada para a aplicação. É executado como um processo isolado no sistema operacional hospedeiro, compartilhando o kernel com outros recipientes. -> Isso corresponde ao Docker, que é uma plataforma que permite a criação, implantação e execução de aplicativos em contêineres.

Portanto, a Letra E é a resposta correta, com cada definição correspondendo ao Kubernetes, Content Delivery Network (CDN) e Docker, respectivamente.

Gabarito: Letra E

---

18. (FCC – TRT - 14ª Região (RO e AC) – 2022) Para coordenar e agendar vários contêineres, habilitar comunicações entre eles e escalar instâncias de contêiner, um analista pode utilizar o

a) Kubernetes, solução utilizada para vários tipos de contêiner, incluindo Docker.
b) Swagger, solução exclusiva para contêineres Docker.
c) Kubernetes, solução exclusiva para contêineres Docker.
d) Swagger, solução exclusiva para contêineres Zuul.
e) Keycloak, solução utilizada para qualquer tipo de contêiner.

Comentários:

O Kubernetes é uma plataforma de código aberto que é usada para automatizar a implantação, o dimensionamento e a operação de aplicativos em contêineres. Ele lida com várias tarefas relacionadas a contêineres, como coordenação, escalonamento, comunicação entre os contêineres e gerenciamento de instâncias de contêineres. Ele não é exclusivo para contêineres Docker, embora seja muito comum o uso do Docker como runtime para os contêineres no ecossistema Kubernetes. O Kubernetes permite a gestão de contêineres de diversas origens.

As outras opções não são corretas porque, vamos analisá-las:

b) Swagger é uma ferramenta usada para definir, construir, documentar e consumir APIs RESTful, não é uma ferramenta específica para coordenação e orquestração de contêineres.

c) Kubernetes é a resposta correta, mas a definição "solução exclusiva para contêineres Docker" não está correta. Kubernetes é capaz de gerenciar contêineres de várias origens, não apenas Docker.

d) Swagger não está relacionado a contêineres Zuul.

e) Keycloak é uma plataforma de gerenciamento de identidade e acesso, não é uma ferramenta para coordenação e orquestração de contêineres.

Portanto, a Letra A é a resposta correta, pois o Kubernetes é a solução adequada para coordenar e agendar vários contêineres, habilitar comunicações entre eles e escalar instâncias de contêiner.

Gabarito: Letra A

---

19. (CESPE – TRT - 8ª Região (PA e AP) – 2022) Em Kubernets, o componente que gerencia os pods que foram criados e que está sem nenhum nó atribuído é o

a) kube-scheduler.
b) kube-apiserver.
c) etcd.
d) kube-controller-manager.
e) cloud-controller-manager.

Comentários:

O componente do Kubernetes que gerencia os Pods que foram criados e que ainda não possuem nenhum nó atribuído é o kube-scheduler. O kube-scheduler é responsável por tomar decisões sobre quais nós os Pods devem ser agendados com base em critérios como requisitos de recursos, afinidades, anti-afinidades e restrições. Ele faz parte do processo de agendamento do Kubernetes, garantindo que os Pods sejam alocados nos nós apropriados para melhorar a eficiência, o desempenho e o equilíbrio de carga no cluster. Portanto, a resposta correta é a opção a) kube-scheduler.

Gabarito: Letra A

---

20. (CESPE – TRT - 8ª Região (PA e AP) – 2022) Entre as funcionalidades do Kubernetes, a que gerencia os contêineres quanto à verificação de integridade definida pelo usuário é

a) gerenciamento de configuração e de segredos.
b) orquestração de armazenamento.
c) lançamentos e reversões automatizadas.
d) empacotamento binário automático.
e) autocorreção.

Comentários:

A funcionalidade do Kubernetes que gerencia os contêineres quanto à verificação de integridade definida pelo usuário é a opção e) autocorreção.

O recurso de autocorreção no Kubernetes é responsável por monitorar o estado dos Pods e dos contêineres em um cluster. Caso um Pod ou contêiner falhe ou não responda às verificações de integridade definidas pelo usuário, o Kubernetes tomará ações para corrigir automaticamente a situação. Isso pode incluir o reinício de contêineres com falha, a substituição de contêineres com problemas e até mesmo a eliminação de Pods não saudáveis. Portanto, a resposta correta é a opção e) autocorreção. Esse recurso contribui para a alta disponibilidade e confiabilidade dos aplicativos implantados no Kubernetes.

Gabarito: Letra E

---

21. (CESPE – BANRISUL – 2022) Em relação a contêineres em aplicações, julgue o item a seguir.

O Kubernetes faz o escalonamento e a recuperação no caso de falha de uma aplicação.

Comentários:

O Kubernetes é uma plataforma de orquestração de contêineres que abrange funcionalidades essenciais, como escalonamento automático e recuperação em caso de falha de aplicativos.

No que diz respeito ao Escalonamento Automático, o Kubernetes oferece a possibilidade de definir políticas de escalonamento automático para os Pods de um aplicativo. Isso implica que, baseado em métricas de uso de recursos, como CPU ou memória, o Kubernetes pode ajustar automaticamente o número de réplicas (instâncias) de um Pod. Esse mecanismo permite que a aplicação se adapte às flutuações de demanda, aumentando ou diminuindo recursos conforme necessário. Isso resulta em uma alocação de recursos mais eficiente e ajuda a garantir que o aplicativo possa enfrentar cargas elevadas sem comprometer sua performance.

Além disso, o Kubernetes também é capaz de realizar a Recuperação de Falhas de forma automática e proativa. A plataforma monitora constantemente o estado dos Pods e dos contêineres em um cluster. Se um Pod ou contêiner apresentar falhas, o Kubernetes age imediatamente para reiniciar o contêiner ou substituir o Pod defeituoso. Isso significa que o aplicativo pode se manter operacional mesmo diante de imprevistos, minimizando o tempo de inatividade. Esse aspecto é crucial para manter a alta disponibilidade dos serviços, assegurando que o aplicativo continue funcionando conforme o planejado, mesmo em cenários de falhas.

Assim, a afirmação está correta ao enfatizar que o Kubernetes oferece funcionalidades de escalonamento automático e recuperação de falhas, contribuindo para a sustentação da disponibilidade e do desempenho dos aplicativos hospedados no cluster.

Gabarito: Correto

---

22. (FGV – FUNSAÚDE-CE – 2021) Pod é uma unidade atômica de escalonamento, implantação e isolamento na execução de um grupo de contêineres no Kubernetes, analise as afirmativas a seguir.

I. Um Pod garante uma mesma localização para os seus contêineres, graças a isso eles têm diversas formas de interagir com bom desempenho, por exemplo, através de troca de arquivos, uso de interface de redes ou de mecanismos de comunicação entre processos.
II. Um Pod tem um endereço IP, um nome e uma faixa de portas compartilhadas por todos os contêineres pertencentes a ele. Isso significa que os contêineres de um mesmo Pod devem ser configurados cuidadosamente a fim de evitar conflitos de portas.
III. Um Pod é um elemento persistente no tempo, ele resiste às operações de redimensionamento, falhas de verificação de sanidade de contêineres e migrações entre nós.

Está correto o que se afirma em:

a) I, somente.
b) II, somente.
c) III, somente.
d) I e II, somente.

e) I e III, somente.

Comentários:

Um Pod é composto por contêineres que compartilham o mesmo endereço IP, nome e intervalo de portas. Portanto, é necessário configurar os contêineres com cuidado para evitar conflitos de portas. Essa abordagem garante que os contêineres dentro de um Pod estejam localizados juntos, permitindo várias maneiras eficientes de interação, como transferência de arquivos, utilização de interfaces de rede e mecanismos de comunicação entre processos.

Vamos analisar cada item: I. Correto. Um Pod é uma unidade lógica que agrupa um ou mais contêineres que compartilham o mesmo espaço de endereçamento e recursos. Isso facilita a interação entre os contêineres, permitindo a troca de arquivos, o uso de interfaces de rede e a comunicação eficiente entre processos.

II. Correto. Um Pod possui um endereço IP único, um nome identificador e uma faixa de portas que é compartilhada por todos os contêineres dentro desse Pod. Isso significa que os contêineres precisam ser configurados de forma adequada para evitar conflitos de portas, já que compartilham a mesma faixa de portas.

III. Incorreto. Um Pod não é uma entidade persistente no tempo. Ele é uma abstração temporária que pode ser escalada, movida entre nós, atualizada e reiniciada. Os Pods não resistem a operações de redimensionamento, falhas de verificação de sanidade de contêineres ou migrações entre nós. Eles podem ser destruídos e recriados conforme necessário.

Gabarito: Letra D

---

23. (CESPE – SEFAZ-CE – 2021) Com a implantação do Kubernetes, é obtido um cluster com pelo menos um nó de trabalho (worker node); os nós de trabalho, por sua vez, hospedam vários componentes da carga de trabalho do aplicativo.

Comentários:

O Kubernetes opera seguindo um modelo de cluster, onde há um ou mais nós de controle (control plane nodes) que gerenciam o estado do cluster e os nós de trabalho (worker nodes) que executam as cargas de trabalho dos aplicativos. Cada nó de trabalho é capaz de executar vários componentes da carga de trabalho, como contêineres, e é escalado conforme necessário para atender à demanda.

Portanto, o gabarito é "Correto" pois reflete com precisão a estrutura básica de um cluster Kubernetes e como os nós de trabalho hospedam os componentes da carga de trabalho do aplicativo.

Gabarito: Correto

---

24. (IADES – BRB – 2021) Kubernetes é uma plataforma de código aberto, portável e extensiva para o gerenciamento de cargas de trabalho e serviços distribuídos em contêineres, que facilita tanto a configuração declarativa quanto a automação. Ele possui um ecossistema grande e de rápido crescimento. Serviços, suporte e ferramentas para Kubernetes estão amplamente disponíveis.
Disponível em: <https://kubernetes.io/pt-br/docs/concepts/overview/>. Acesso em: 25 jun. 2021, com adaptações.

Com base no texto apresentado e considerando o contexto do Kubernetes, assinale a alternativa que corresponde à ferramenta disponibilizada para realizar operações nos clusters Kubernetes, por meio de interface de linha de comando, pela qual é possível realizar a implantação de aplicações, inspecionar e gerenciar recursos do cluster e visualizar logs.

a) Kubeadm
b) Minikube
c) Kubectl
d) Kind
e) Kubelet

Comentários:

A ferramenta disponibilizada para realizar operações nos clusters Kubernetes por meio de interface de linha de comando, que permite implantar aplicações, inspecionar e gerenciar recursos do cluster, e visualizar logs, é o "kubectl". O "kubectl" é uma ferramenta de linha de comando para interagir com clusters Kubernetes. Ela é usada para gerenciar e operar recursos dentro do cluster. Com o "kubectl", os administradores, desenvolvedores e engenheiros podem executar uma variedade de tarefas, desde a implantação de aplicativos até a observação e ajuste do estado do cluster. Portanto, a alternativa correta é a letra "c) Kubectl".

Gabarito: Letra C

---

25. (CESPE – SEFAZ-CE – 2021) No Kubernetes, kubelet é uma pequena aplicação localizada em um nó que se comunica com o plano de controle, assegurando que os containers estejam em execução em um pod, que consiste no menor e mais simples objeto do Kubernetes.

Comentários:

O "kubelet" é uma pequena aplicação localizada em cada nó do cluster e é responsável por garantir que os contêineres estejam em execução nos pods, de acordo com as especificações definidas. O "kubelet" monitora os pods atribuídos ao nó em que está sendo executado. Ele se comunica com o plano de controle do Kubernetes para receber as informações sobre quais pods devem ser executados em cada nó, além de reportar o status e a saúde dos pods de volta para o plano de controle. Isso inclui monitorar se os contêineres estão em execução, reiniciando-os se necessário e realizando outras ações de gerenciamento.

É importante notar que o pod é realmente a unidade mais básica no Kubernetes. Um pod é um agrupamento de um ou mais contêineres, compartilhando o mesmo espaço de rede e armazenamento. O "kubelet" garante que esses contêineres dentro de um pod estejam em execução e saudáveis.

Gabarito: Correto

---

26. (FGV – TJ-RO – 2021) A equipe de desenvolvimento de sistemas de um tribunal de contas está guiando a implantação de um Webservice REST. A implantação será dividida nos seguintes grupos distintos de containers Docker:

- Grupo A: responsável pela execução da aplicação do Webservice REST
- Grupo B: responsável pela execução do Sistema Gerenciador de Banco de Dados utilizado pelo
Webservice REST Os Grupos A e B terão seu próprio contexto de armazenamento e rede a serem orquestrados por um cluster de Kubernetes.

Para que a conexão de rede entre os containers dos Grupos A e B seja bem-sucedida pelo orquestrador, independentemente dos endereços IP a eles atribuídos, deverá ser configurado um novo:

a) pod
b) replicaSet
c) service
d) ingress
e) statefulSet

Comentários:

O cenário descrito envolve a implantação de um Webservice REST dividido em dois grupos de containers: Grupo A para a aplicação do Webservice REST e Grupo B para o Sistema Gerenciador de Banco de Dados. Para que esses grupos de containers possam se comunicar entre si de forma eficiente e independente dos endereços IP atribuídos a eles, é necessário configurar um

mecanismo que permita a descoberta dinâmica de serviços e que gerencie o tráfego entre os containers.

A opção correta é a letra c) "service".

O Kubernetes oferece o recurso de "Service" para resolver exatamente esse tipo de cenário. Um "Service" é um objeto abstrato que define uma política de acesso à rede para um conjunto de pods. Ele permite que os pods dentro de um grupo, como o Grupo A e o Grupo B neste caso, possam se comunicar entre si usando um nome DNS, independentemente dos IPs individuais dos pods. O "Service" age como um balanceador de carga interno, distribuindo o tráfego entre os pods correspondentes.

Ao configurar um "Service", você cria um ponto de acesso único para o grupo de pods, permitindo que eles se comuniquem de maneira confiável e escalável. O "Service" também pode ser configurado para expor os pods para o tráfego externo, se necessário.

Portanto, a alternativa correta é a letra c) "service". Ela é a escolha apropriada para estabelecer uma conexão de rede bem-sucedida entre os containers dos Grupos A e B, garantindo que eles possam se comunicar eficientemente no contexto de um cluster Kubernetes.

Gabarito: Letra C

---

27. **(CESPE – SERPRO – 2020)** A camada de gerenciamento do Kubernetes possui o componente etcd, cuja função é observar pods que foram criados sem nenhum node atribuído e selecionar um node para execução.

Comentários:

O etcd é a base de dados distribuída utilizada pelo Kubernetes para armazenar todo o estado do cluster, incluindo informações sobre os recursos, configurações e eventos. Ele não tem a função de observar pods sem nenhum node atribuído e selecionar um node para execução. A atribuição de pods a nós de trabalho (nodes) é uma responsabilidade do componente de agendamento (scheduler) do Kubernetes.

O etcd é uma parte crítica do sistema, fornecendo consistência e persistência para todas as informações do cluster, permitindo que o Kubernetes mantenha um registro confiável do estado atual e desejado dos recursos. Portanto, o gabarito é "Errado", pois o etcd não desempenha a função mencionada.

Gabarito: Errado

LISTA DE QUESTÕES

1. (CESPE – SERPRO– 2023) Julgue o item subsequente, referentes a ferramentas de integração assíncrona e contêineres.

   O Kubernetes permite agrupar hosts executados em contêineres Linux (LXC) em clusters Kubernetes, os quais podem conter um kubelet, que é um grupo de um ou mais contêineres implantados em um nó, suportando aplicações que realizam, por exemplo, a transmissão de dados em tempo real pelo Apache Kafka.

2. (FGV – SEF MG – 2023) Várias plataformas FaaS de código aberto foram desenvolvidas como alternativa às soluções serverless dos provedores de nuvem pública. As plataformas FaaS de código aberto são quase exclusivamente implementadas sobre o Kubernetes, como por exemplo o OpenFaaS.

   Em relação às características que possuem em comum os sistemas FaaS de código aberto baseados em kubernetes, assinale a afirmativa incorreta.

   a) As invocações às funções são roteadas por um gateway, definido pelo sistema FaaS ou por uma instância Kubernetes Ingress.
   b) O balanceamento de carga das requisições entre instâncias de função é implementado aproveitando as instâncias de serviço Kubernetes.
   c) Para o autoescalonamento, a maioria dos sistemas FaaS usa o Horizontal Pod Autoscaling (HPA) integrado do Kubernetes, que implementa o autoescalonamento baseado em recursos de computação das instâncias das funções.
   d) Para o autoescalonamento, mecanismos de escalonamento personalizados também podem ser implementados, como escalonamento automático com base no número de solicitações simultâneas em andamento.
   e) Para monitorar as instâncias de função, a maioria dos sistemas FaaS de código aberto utiliza suas próprias soluções de captura de métricas e visualização.

3. (FGV – SEF MG– 2023) Kubernetes é um sistema de orquestração de contêineres opensource que automatiza a implantação, o dimensionamento e a gestão de aplicações em contêineres.

   Em relação ao conceito de Kubernetes, assinale a afirmativa incorreta.

   a) Kube-proxy é um proxy de rede executado e mantém regras de rede em cada máquina do cluster
   b) Kubernetes utilizam controladores que rastreiam pelo menos um tipo de recurso Kubernetes.

c) Kubernetes utiliza contexts como mecanismo para isolar grupos de recursos dentro de um único cluster.
d) Kubelet é um agente que é executado em cada máquina do cluster; ele garante que os contêineres estejam sendo executados em um Pod.
e) Os objetos do Kubernetes são entidades persistentes no Kubernetes e utilizam estas entidades para representar o estado do cluster.

4. (UFU – MG – UFU – MG – 2023) Kubernetes é um plataforma de código aberto, portável e extensiva, utilizada para automatizar a implantação, o dimensionamento e o gerenciamento de cargas de trabalho e serviços distribuídos em contêineres. O cluster Kubernetes consiste em um conjunto de servidores de processamento que executam aplicações conteinerizadas. Esses servidores hospedam Pods, que são componentes de uma aplicação. Considerando as características e a operacionalização do Kubernetes, analise as asserções abaixo.

I. No Kubernetes, as configurações de contexto de segurança especificadas para um POD não se aplicam a todos os Contêineres no POD, mas somente aos referidos no arquivo de configuração do POD.

II. A linha de comando "kubectl exec -it POD_NAME – sh" é utilizada para inicializar um agrupamento de contêineres definido em um POD denominado POD_NAME.

III. O Kubernetes fornece diferentes níveis de Qualidade de Serviço aos PODs por meio da classificação de cada POD em uma classe QoS específica. Três classes denominadas como "Burstable", "Guaranteed" e "BestEffort são definidas com base nas solicitações de recursos e nos limites de recursos estabelecidos para os Contêineres do POD.

IV. O Kubernetes utiliza as classes QoS para tomar decisões sobre quais PODs despejar quando não houver recursos suficientes disponíveis em um nó. Em situações de falta de recursos, o Kubernetes interromperá a execução de PODs de acordo com a classificação atribuída, sendo primeiro removidos aqueles classificados como "BestEffort", seguido pelos "Guaranteed", e por fim, os "Burstable".

Marque a alternativa que classifica corretamente as asserções como verdadeiras (V) ou falsas (F)

a) I - V; II - V; III - F; IV - F.
b) I - F; II - F; III - V; IV - F.
c) I - V; II - F; III - F; IV - V.
d) I - F; II - V; III - V; IV - V.

5. (CESPE – SEFIN de Fortaleza - CE – 2023) Tendo em vista que, no atual cenário de desenvolvimento de aplicações web, é essencial considerar princípios, como consistência e

escalabilidade, e práticas, como automação do processo de implantação e integração do código-fonte, julgue o item subsequente.

As ferramentas DevOps incluem o Kubernetes, uma plataforma de orquestração de contêineres que permite gerenciar e escalonar aplicativos em contêineres em diferentes ambientes e pode ser integrada a outras ferramentas DevOps, como Docker e Jenkins.

6. (INSTITUTO AOCP – IF-MA – 2023) No Kubernetes, qual é a abordagem correta para implementar comunicação segura entre os componentes do cluster e garantir autenticação e autorização adequadas?

a) Desabilitar o controle de acesso baseado em função (RBAC) e permitir o acesso a todos os usuários e serviços.
b) Utilizar certificados autoassinados para todos os componentes do cluster e ignorar a validação de certificados.
c) Implementar políticas de segurança usando o etcd como armazenamento centralizado de credenciais e chaves de criptografia.
d) Configurar o Kubernetes para utilizar criptografia TLS, certificados válidos e controle de acesso baseado em função (RBAC).
e) Utilizar somente redes não criptografadas entre os componentes do cluster para melhorar o desempenho.

7. (FGV – MPE-GO – 2022) Uma equipe de especialistas em infraestrutura está implantando o orquestrador Kubernetes em um servidor do MP-GO que possui vários containers Docker. Foi decidido que alguns desses containers serão agrupados na unidade básica de operações do Kubernetes.

Para isso, a equipe deve definir um novo

a) statefulSet.
b) replicaSet.
c) service.
d) ingress.
e) pod.

8. (FEPESE – FAPESC – 2022) A utilização de Kubernetes no gerenciamento de aplicações está sendo amplamente difundida.

Com relação ao Kubernetes, é correto afirmar:

a) O Kubernetes não realiza o escalonamento e a recuperação no caso de falhas.
b) Os fluxos de trabalho de integração contínua, entrega e implantação (CI/CD) são determinados pelo Kubernetes.

c) Fornece serviços em nível de aplicação, tais como middlewares, bancos de dados e caches.
d) Limita o tipo de aplicações executadas a aplicações statefull.
e) Se o tráfego para um contêiner gerenciado pelo Kubernetes for alto, ele pode balancear a carga e distribuir o tráfego de rede para que a implantação seja

9. (FGV – MPE-GO – 2022) Uma equipe de especialistas em infraestrutura está implantando o orquestrador Kubernetes em um servidor do MP-GO que possui vários containers Docker. Foi decidido que alguns desses containers serão agrupados na unidade básica de operações do Kubernetes.

Para isso, a equipe deve definir um novo

a) statefulSet.
b) replicaSet.
c) service.
d) ingress.
e) pod.

10. (FEPESE – Polícia Científica - SC – 2022) Assinale a alternativa que define corretamente o Kubernetes.

a) É um tipo de malware que abre uma ou mais portas para acesso remoto de modo a executar código ou aplicações maliciosas.
b) É uma arquitetura de nuvem pública que possibilita a virtualização de infraestrutura e o seu provimento como serviço (Sas).
c) É uma técnica de Phishing na qual a vítima é induzida a prover dados confidenciais como senhas utilizando-se de engenharia social.
d) É um Desktop Virtual que provê um ambiente seguro e monitorado como estação de trabalho para utilização em home-office.
e) É um orquestrador que possibilita automatizar a implantação e escala de aplicações em containers.

11. (FGV – TJ-DFT – 2022) A analista Ana precisa implantar o pod LogPod no cluster de Kubernetes KCluster do TJDFT de forma que todos os nós elegíveis do KCluster executem uma cópia do LogPod.

Para que o KCluster apresente uma cópia do LogPod em cada nó elegível, de forma automática, Ana deve implantar o LogPod utilizando o recurso do Kubernetes:

a) ReplicationController;
b) ReplicaSet;
c) StatefulSet;
d) DaemonSet;

e) EndpointSlice.

12. (UFMT – POLITEC-MT – 2022) O Kubernetes é utilizado para automatizar o gerenciamento de aplicativos em contêiner. A respeito do Kubernetes, marque V para as afirmativas verdadeiras e F para as falsas.

( ) Utiliza pod para criar um ambiente de teste local.
( ) Usa volumes como máquinas de processamento.
( ) Tem kubectl como interface de linha de comando.
( ) Trata-se de produto comercial, de código fechado.

Assinale a sequência correta.

a) V, F, V, F
b) F, V, V, V
c) V, V, F, V
d) V, F, F, F
e) F, F, V, F

13. (FGV – SENADO– 2022) Uma instituição acadêmica está desenvolvendo um ambiente de laboratório virtual em kubernetes, com alta disponibilidade. O sistema provisiona automaticamente um novo pod quando um aluno se autentica.
Um dos pré-requisitos do projeto é isolar os pods dos alunos em um pool de nodes exclusivo em alta disponibilidade.
Assinale a opção que indica a funcionalidade do kubernetes adequada a esse pré-requisito.

a) Ingress
b) Node affinity
c) Limit Ranges
d) Pod Overhead
e) Node-pressure Eviction

14. (FGV – TRT - 13ª Região (PB) – 2022) A infraestrutura recomendada para instalar o Rancher em um cluster Kubernetes K3 de alta disponibilidade é

a) dois nós Linux, normalmente máquina virtual, e um banco de dados externo do tipo NoSQL.
b) um cluster com Linux com menos dois nós ou quatro nós se Windows Server normalmente máquinas virtuais; com pelos menos 64 GB de memória RAM no provedor de infraestrutura e um banco de dados externo do tipo NoSQL.
c) um cluster com Linux de dois nós normalmente máquinas virtuais; com pelos menos 48 GB de memória RAM no provedor de infraestrutura.

d) dois nós Linux, normalmente máquinas virtuais; um banco de dados externo do tipo relacional, um balanceador de carga e um registro DNS.
e)
quatro nós Windows Server, normalmente máquinas virtuais; um banco de dados externo do tipo relacional, um balanceador de carga entre os quatro nós e dois registros DNS como redundância.

15. (FCC – TRT - 19ª Região (AL)– 2022) Nos projetos da Plataforma Digital do Poder Judiciário PDPJ-Br,

a) os ambientes de deployment estão instalados em clusters Kubernetes, gerenciados por meio da ferramenta open source de orquestração de containers Rancher.
b) a aplicação deve possuir os arquivos yanl, para que o deploy ocorra de forma automática por meio de pipelines de CI/CD (Continuous Delivery e Continuous Integration) executados pelo GitLab do STF (Supremo Tribunal Federal).
c) a aplicação deve possuir um arquivo Rancherfile, especificando o modo pelo qual ela deve ser empacotada.
d) deve haver o arquivo .gitlab-ci.xml definindo a pipeline de deploy automático.
e) deve haver, na pasta kubernets, o arquivo de configuração do ingress da aplicação, que é obrigatório.

16. (FCC – TRT - 22ª Região (PI) – 2022) No Kubernetes, os contêineres gerenciados pelo kubelet podem usar a estrutura de hook do ciclo de vida do contêiner para executar código acionado por eventos durante o gerenciamento de seu ciclo de vida. Há dois tipos de hooks:

I. Este hook é executado imediatamente após um contêiner ser criado. Mas não há garantia de que o hook será executado antes do ENTRYPOINT do contêiner. Nenhum parâmetro é passado para o handler.

II. Esse hook é chamado imediatamente antes de um contêiner ser terminated devido a uma solicitação de API ou um gerenciamento de evento como liveness/startup probe failure, preemption, resource contention e outros. Uma chamada a este hook falha se o contêiner já está em um estado terminated ou completed e o hook deve ser concluído antes que o sinal TERM seja enviado para parar o contêiner. Nenhum parâmetro é passado para o handler.

Os hooks I e II são, correta e respectivamente,
a) PostStart e PreStop.
b) HookStart e HookStop.
c) PodStart e PodStop.
d) KubeStart e KubeStop.
e) KubeletStart e KubeletStop.

17. (FCC – TRT - 17ª Região (ES) – 2022) Considere as seguintes definições:

I. Gerencia os containers em execução e, por isso, ele também é chamado de Orquestrador de Containers. Através dele se pode definir o estado de um sistema completo, por exemplo, baseado em Microservices, seguindo boas práticas de infraestrutura como código, permitindo balanceamento de carga, alta disponibilidade, atualizações em lote e rollbacks.

II. Refere-se a um grupo de servidores distribuídos geograficamente que trabalham em conjunto para oferecer uma rápida entrega de conteúdo de internet. Permite uma transferência rápida dos ativos necessários para carregar o conteúdo da internet, incluindo páginas HTML, arquivos em JavaScript, folhas de estilo, imagens e vídeos.

III. Projeto opensource escrito em Go, que torna a criação e o gerenciamento de containers Linux mais fácil. O container é construído usando namepspaces, cgroups, chroot entre outras funcionalidades do kernel para criar uma área isolada para a aplicação. É executado como um processo isolado no sistema operacional hospedeiro, compartilhando o kernel com outros recipientes.

As definições I, II e III correspondem, correta e respectivamente, a

a) Kubernetes, Docker e Content Delivery Network (CDN).
b) Content Delivery Network (CDN), Kubernetes e Docker.
c) Content Delivery Network (CDN), Docker e Kubernetes.
d) Docker, Content Delivery Network (CDN) e Kubernetes.
e) Kubernetes, Content Delivery Network (CDN) e Docker.

18. (FCC – TRT - 14ª Região (RO e AC) – 2022) Para coordenar e agendar vários contêineres, habilitar comunicações entre eles e escalar instâncias de contêiner, um analista pode utilizar o

a) Kubernetes, solução utilizada para vários tipos de contêiner, incluindo Docker.
b) Swagger, solução exclusiva para contêineres Docker.
c) Kubernetes, solução exclusiva para contêineres Docker.
d) Swagger, solução exclusiva para contêineres Zuul.
e) Keycloak, solução utilizada para qualquer tipo de contêiner.

19. (CESPE – TRT - 8ª Região (PA e AP) – 2022) Em Kubernets, o componente que gerencia os pods que foram criados e que está sem nenhum nó atribuído é o

a) kube-scheduler.
b) kube-apiserver.
c) etcd.
d) kube-controller-manager.
e) cloud-controller-manager.

20. (CESPE – TRT - 8ª Região (PA e AP) – 2022) Entre as funcionalidades do Kubernetes, a que gerencia os contêineres quanto à verificação de integridade definida pelo usuário é

a) gerenciamento de configuração e de segredos.
b) orquestração de armazenamento.
c) lançamentos e reversões automatizadas.
d) empacotamento binário automático.
e) autocorreção.

21. (CESPE – BANRISUL – 2022) Em relação a contêineres em aplicações, julgue o item a seguir.

O Kubernetes faz o escalonamento e a recuperação no caso de falha de uma aplicação.

22. (FGV – FUNSAÚDE-CE – 2021) Pod é uma unidade atômica de escalonamento, implantação e isolamento na execução de um grupo de contêineres no Kubernetes, analise as afirmativas a seguir.

I. Um Pod garante uma mesma localização para os seus contêineres, graças a isso eles têm diversas formas de interagir com bom desempenho, por exemplo, através de troca de arquivos, uso de interface de redes ou de mecanismos de comunicação entre processos.
II. Um Pod tem um endereço IP, um nome e uma faixa de portas compartilhadas por todos os contêineres pertencentes a ele. Isso significa que os contêineres de um mesmo Pod devem ser configurados cuidadosamente a fim de evitar conflitos de portas.
III. Um Pod é um elemento persistente no tempo, ele resiste às operações de redimensionamento, falhas de verificação de sanidade de contêineres e migrações entre nós.

Está correto o que se afirma em:

a) I, somente.
b) II, somente.
c) III, somente.
d) I e II, somente.
e) I e III, somente.

23. (CESPE – SEFAZ-CE – 2021) Com a implantação do Kubernetes, é obtido um cluster com pelo menos um nó de trabalho (worker node); os nós de trabalho, por sua vez, hospedam vários componentes da carga de trabalho do aplicativo.

24. (IADES – BRB – 2021) Kubernetes é uma plataforma de código aberto, portável e extensiva para o gerenciamento de cargas de trabalho e serviços distribuídos em contêineres, que facilita tanto a configuração declarativa quanto a automação. Ele possui um ecossistema grande e de

rápido crescimento. Serviços, suporte e ferramentas para Kubernetes estão amplamente disponíveis.
Disponível em: <https://kubernetes.io/pt-br/docs/concepts/overview/>. Acesso em: 25 jun. 2021, com adaptações.

Com base no texto apresentado e considerando o contexto do Kubernetes, assinale a alternativa que corresponde à ferramenta disponibilizada para realizar operações nos clusters Kubernetes, por meio de interface de linha de comando, pela qual é possível realizar a implantação de aplicações, inspecionar e gerenciar recursos do cluster e visualizar logs.

a) Kubeadm
b) Minikube
c) Kubectl
d) Kind
e) Kubelet

25. (CESPE – SEFAZ-CE – 2021) No Kubernetes, kubelet é uma pequena aplicação localizada em um nó que se comunica com o plano de controle, assegurando que os containers estejam em execução em um pod, que consiste no menor e mais simples objeto do Kubernetes.

26. (FGV – TJ-RO – 2021) A equipe de desenvolvimento de sistemas de um tribunal de contas está guiando a implantação de um Webservice REST. A implantação será dividida nos seguintes grupos distintos de containers Docker:

- Grupo A: responsável pela execução da aplicação do Webservice REST
- Grupo B: responsável pela execução do Sistema Gerenciador de Banco de Dados utilizado pelo
Webservice REST Os Grupos A e B terão seu próprio contexto de armazenamento e rede a serem orquestrados por um cluster de Kubernetes.

Para que a conexão de rede entre os containers dos Grupos A e B seja bem-sucedida pelo orquestrador, independentemente dos endereços IP a eles atribuídos, deverá ser configurado um novo:

a) pod
b) replicaSet
c) service
d) ingress
e) statefulSet

27. (CESPE – SERPRO – 2020) A camada de gerenciamento do Kubernetes possui o componente etcd, cuja função é observar pods que foram criados sem nenhum node atribuído e selecionar um node para execução.

GABARITO

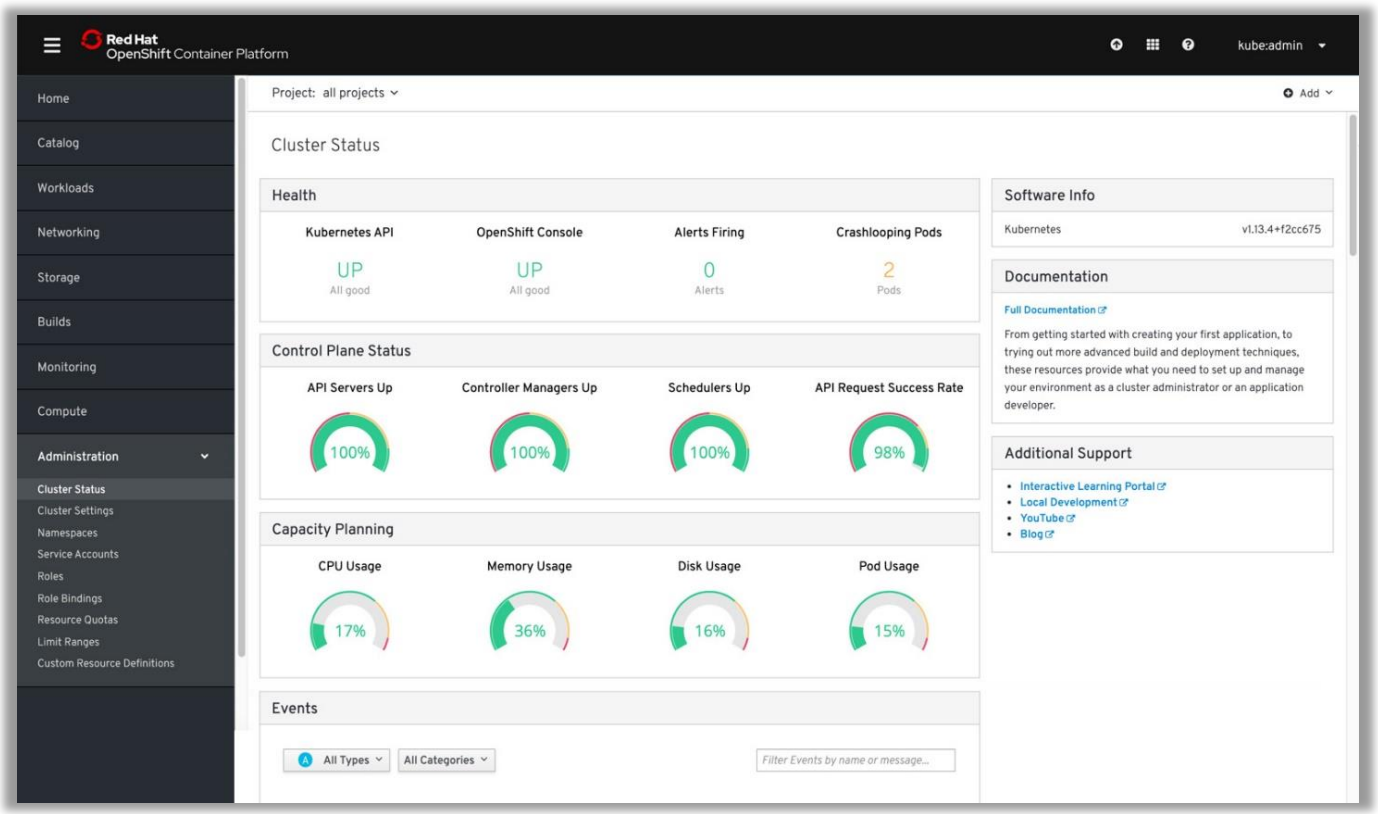
1. Errado
2. Errado
3. Letra C
4. Letra B
5. Correto
6. Letra D
7. Letra E
8. Letra E
9. Letra E
10. Letra E
11. Letra D
12. Letra E
13. Letra B
14. Letra D
15. Letra A
16. Letra A
17. Letra E
18. Letra A
19. Letra A
20. Letra E
21. Correto
22. Letra D
23. Correto
24. Letra C
25. Correto
26. Letra C
27. Errado

OpenShift

Conceitos Básicos

INCIDÊNCIA EM PROVA: MÉDIA

O OpenShift é uma ferramenta muito poderosa que além de **orquestrar e gerenciar containers, oferece segurança, monitoramento e controle**. Com o OpenShift podemos criar um ambiente baseado em containers e colocar aplicações construídas em diversas linguagens como: Java, Python, Ruby, e PHP, para rodar e escalar facilmente, trazendo maior agilidade no desenvolvimento utilizando metodologias DevOps, atendendo mais rapidamente e com menor esforço as demandas de negócio.



A containerização é fator fundamental aqui no Openshift, já que ele monitora os status dos clusters. Se lembrar do funcionamento do Kubernete vai perceber que temos containers como serviço basicamente, que em essência nada mais é que a automatização do dimensionamento das operações com as aplicações. Obviamente também temos automonitoramento, balanceamento, orquestração, armazenamento, etc.

Dito isso, no OpenShift a Red Hat buscou ampliar esse conceito, **enquanto que o Kubernetes é o Kernel dos sistemas distribuídos em containers, o Openshift é uma plataforma de containers kubernetes baseada em nuvem**, literalmente um legitimo PAAS (Plataform as a service), além de parcialmente ser construído em Docker também.

Logo, o Openshift oferece **segurança consistente, monitoramento integrado, gerenciamento centralizado de políticas e compatibilidade com cargas de trabalho de contêiner do Kubernetes**. É rápido, permite o provisionamento de autoatendimento e se integra a uma variedade de ferramentas. Em outras palavras, não há dependência de fornecedor.

Tanto o Kubernetes quanto o OpenShift apresentam uma arquitetura robusta e escalável que permite o desenvolvimento, a implantação e o gerenciamento de aplicativos rápidos e em larga escala. Temos então algumas diferenças fundamentais aqui, principalmente no tocante a implantação, vejamos:

O Kubernetes oferece mais flexibilidade como uma estrutura de código aberto e pode ser instalado em praticamente qualquer plataforma — como Microsoft Azure e AWS — bem como em qualquer distribuição Linux, incluindo Ubuntu e Debian. O OpenShift, por outro lado, requer o Red Hat Enterprise Linux Atomic Host (RHELAH), Fedora ou CentOS proprietário da Red Hat. Isso restringe as opções para muitas empresas, especialmente se elas ainda não estiverem usando essas plataformas. **As imagens no Openshift são armazenadas no componente registry**.

Enquanto que o Kubernetes contém uma interface web complexa que pode confundir os novatos. Os usuários que desejam acessar a interface gráfica do usuário (GUI) da Web do Kubernetes devem instalar o painel do Kubernetes e usar o kube-proxy para enviar a porta de sua máquina para o servidor de cluster. Já o OpenShift, por outro lado, apresenta um console web intuitivo que inclui uma página de login com um toque. O console oferece uma interface simples e baseada em formulário, permitindo que os usuários adicionem, excluam e modifiquem recursos. As imagens dos contêineres no Openshift podem ser armazenadas no componente denominado registry. No Openshift o estado persistente do master é armazenado no componente etcd.

Logo, tanto o Kubernetes quanto o OpenShift são sistemas populares de gerenciamento de contêineres e cada um tem seus recursos e benefícios exclusivos. Enquanto o Kubernetes ajuda a automatizar a implantação, o dimensionamento e as operações de aplicativos, o OpenShift é a plataforma de contêiner que funciona com o Kubernetes para ajudar os aplicativos a serem executados com mais eficiência.

QUESTÕES COMENTADAS – OPENSHIFT

1. **(CESPE / TJPA – 2020)** O Openshift provê recursos a partir do kubernets, sendo capaz de executar e disponibilizar aplicações a partir de contêineres. As imagens dos contêineres no Openshift podem ser armazenadas no componente denominado

   a) Pod
   b) Build
   c) Secret
   d) Registry
   e) Master

Comentários:

As imagens dos contêineres no Openshift conforme vimos são armazenadas no Registry.

Gabarito: Letra D

---

2. **(FGV / FUNSAÚDE-CE – 2021)** Em um *cluster* Openshift, há uma série de configurações que são feitas e devem ser persistidas. O estado persistente do master é armazenado no componente

   a) etcd
   b) haproxy
   c) API server
   d) Namespace
   e) Replica controller

Comentários:

Conforme vimos, no Openshift o estado persistente do master é armazenado no componente etcd.

Gabarito: Letra A

---

3. **(CESPE / ME – 2020)** Na situação em que vários aplicativos desenvolvidos e baseados em microsserviços com abordagem de containers devem ser disponibilizados aos usuários de uma organização, é correto instalar e configurar a OpenShift Container Platform, que é a solução de gerenciamento de containers da RedHat, pois não há solução da Microsoft que oferte serviços de Kubernetes para disponibilização desses aplicativos.

Comentários:

A questão não faz nenhum sentido, Kubernetes pode ser utilizado em qualquer plataforma, na Microsoft é oferecido através do AKS (Serviço de Kubernetes da Azure) no restante sobre OpenShift está correto.

Gabarito: Errado

LISTA DE QUESTÕES – OPENSHIFT

1. **(CESPE / TJPA – 2020)** O Openshift provê recursos a partir do kubernets, sendo capaz de executar e disponibilizar aplicações a partir de contêineres. As imagens dos contêineres no Openshift podem ser armazenadas no componente denominado

   a) Pod
   b) Build
   c) Secret
   d) Registry
   e) Master

2. **(FGV / FUNSAÚDE-CE – 2021)** Em um *cluster* Openshift, há uma série de configurações que são feitas e devem ser persistidas. O estado persistente do master é armazenado no componente

   a) etcd
   b) haproxy
   c) API server
   d) Namespace
   e) Replica controller

3. **(CESPE / ME – 2020)** Na situação em que vários aplicativos desenvolvidos e baseados em microsserviços com abordagem de containers devem ser disponibilizados aos usuários de uma organização, é correto instalar e configurar a OpenShift Container Platform, que é a solução de gerenciamento de containers da RedHat, pois não há solução da Microsoft que oferte serviços de Kubernetes para disponibilização desses aplicativos.

GABARITO – OPENSHIFT

1. ERRADO
2. LETRA A
3. LETRA D

Sumário

Apresentação da Aula 2
Rancher 3
Conceitos Básicos 3
Overview 5
Autenticação e Autorização 6
Controle de acesso baseado em função (RBAC) 6
Provisionando Drivers 6
Políticas de segurança de pods 7
Administração de Cluster 7
Cluster Kubernetes K3s de Alta Disponibilidade 9

APRESENTAÇÃO DA AULA

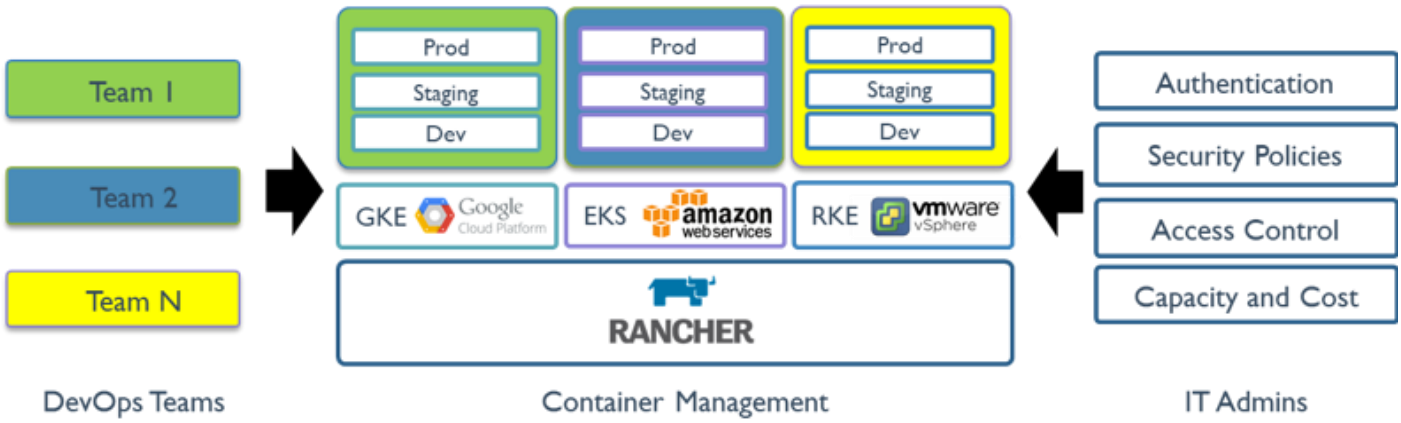
Olá pessoal! Hoje, vamos explorar o mundo do Rancher e entender como essa plataforma tem impactado a forma como lidamos com clusters Kubernetes e sua administração. O Rancher é uma ferramenta essencial que simplifica e aprimora a gestão de clusters, permitindo maior eficiência e controle em todo o processo.

Para começar, vamos discutir os conceitos básicos do Rancher, abrangendo uma visão geral de suas funcionalidades. Em seguida, mergulharemos na autenticação e autorização, destacando como o Rancher gerencia o acesso aos clusters e como o controle de acesso baseado em função (RBAC) desempenha um papel importante nesse contexto.

Um aspecto fundamental do Rancher é a capacidade de provisionar drivers para suportar diferentes ambientes e infraestruturas. Vamos analisar como essa funcionalidade é implementada e como podemos personalizar o Rancher para se adaptar às necessidades específicas de nossa infraestrutura. Não podemos esquecer das políticas de segurança de pods, que realizam a garantia da segurança dos aplicativos em execução nos clusters. Vamos explorar como o Rancher lida com essas políticas e como podemos configurá-las de forma eficaz.

Além disso, falaremos sobre a administração de cluster, abordando as melhores práticas para manter nossos clusters Kubernetes gerenciados pelo Rancher em perfeito funcionamento. Também discutiremos a configuração de clusters Kubernetes K3s de alta disponibilidade, uma parte essencial para garantir a confiabilidade e o desempenho de nossos ambientes.

À medida que avançamos, é importante lembrar que a implementação bem-sucedida do Rancher vai além da adoção de ferramentas e práticas específicas. Requer uma mudança cultural, promovendo a colaboração entre equipes e investindo em treinamento para que todos possam tirar o máximo proveito dessa poderosa plataforma. Vamos mergulhar nos conceitos do Rancher e explorar como ele pode aprimorar nossos ambientes de Kubernetes! 😀



RANCHER

Conceitos Básicos

Rancher é um conjunto completo de software para equipes que adotam containers. Ele lida com os desafios operacionais e de segurança de gerenciar vários clusters em qualquer infraestrutura, ao mesmo tempo em que fornece às equipes de DevOps ferramentas integradas para executar cargas de trabalho containerizadas.

O Rancher é como um gerente que ajuda a cuidar de vários grupos de aplicativos, chamados de clusters, de maneira organizada. Ele não só ajuda a colocar esses clusters em diferentes lugares, como datacenters, nuvens, mas também os faz trabalhar juntos de maneira mais segura e inteligente.

> O Rancher é uma ferramenta de gerenciamento de cluster que ajuda a provisionar, gerenciar e monitorar clusters em qualquer lugar e com qualquer provedor de nuvem ou infraestrutura.

O Rancher fornece uma interface de usuário e uma API comuns para gerenciar clusters de diferentes tipos como Docker Swarm, Mesos, Nomad, Amazon Elastic Container Service (ECS), Google Kubernetes Engine (GKE), Azure Container Service (AKS). Isso permite que as organizações simplifiquem a gestão de seus clusters, independentemente da tecnologia de orquestração de contêineres que estejam usando. No entanto, o Rancher é mais conhecido por seu suporte ao Kubernetes. Ele oferece uma ampla gama de recursos para gerenciar clusters Kubernetes, incluindo:

- Provisionamento de cluster
- Gerenciamento de nós
- Gerenciamento de redes
- Gerenciamento de armazenamento
- Gerenciamento de aplicativos
- Monitoramento de cluster

Pense no Rancher como um assistente que ajuda a criar e organizar grupos de aplicativos especiais chamados de clusters. Ele pode criar esses clusters em um provedor de hospedagem, adicionar máquinas de computação e depois instalar o Kubernetes nelas. Também consegue trazer clusters Kubernetes já existentes que estejam funcionando em qualquer lugar.

O que torna o Rancher ainda mais útil é que ele faz muitas coisas extras para tornar tudo mais fácil. Primeiro, reúne todos os lugares onde as pessoas podem entrar nos clusters e define quem pode fazer o quê, para que tudo fique organizado e seguro.

Além disso, ele permite que você fique de olho em todos os clusters e o que está acontecendo com eles. Se algo estiver errado, ele avisa. E não para por aí, ele também ajuda a enviar os registros dos aplicativos para outros lugares importantes e se conecta com uma "biblioteca" cheia de aplicativos que você pode instalar facilmente.

E se você já usa um sistema especial para construir e testar seus aplicativos, o Rancher pode se conectar a ele. Mas se você ainda não tem um sistema assim, não se preocupe! O Rancher tem uma ferramenta chamada Fleet que ajuda a colocar os aplicativos no lugar certo de forma automática.

Imagine que o Rancher é como um assistente para implantar e proteger esses clusters. Ele pode ajudar a implantá-los em lugares diferentes, como computadores dedicados, nuvens particulares, nuvens públicas ou até mesmo um sistema chamado vSphere. E o melhor é que ele também ajuda a proteger esses aplicativos com regras de segurança globais.

A maioria dos desenvolvedores quer escrever código e criar seus aplicativos de maneira segura de forma que não tenham que se preocupar com a parte técnica complicada. O Rancher faz exatamente isso, tornando simples oferecer o Kubernetes como um serviço para as equipes de desenvolvimento. Eles podem implantar seus trabalhos em containers com um simples clique, seja localmente, na nuvem, de forma híbrida ou em locais específicos. Enquanto isso, os operadores podem manter o controle e a observabilidade [1]de tudo.

O Rancher ajuda a unir os diferentes clusters Kubernetes, oferecendo segurança centralizada, controle de acesso, auditoria, backups, atualizações e alertas. É como ter todas as peças juntas em um quebra-cabeça. Você pode implantar e proteger seus clusters de forma consistente em qualquer lugar, usando uma interface fácil de usar ou comandos poderosos.

Se você tem clusters locais ou em serviços de nuvem, o Rancher cuida de todos eles a partir de um único lugar. Imagine um controle remoto que gerencia tudo. Você pode definir regras de segurança, acompanhar o que está acontecendo e implantar aplicativos em vários clusters com facilidade.

Imagine poder escolher entre várias ferramentas e aplicativos populares de código aberto, tudo em um só lugar. O Rancher permite isso através do Catálogo de Aplicativos. Você pode usar o Helm para implantar essas ferramentas rapidamente, sem perder tempo configurando tudo manualmente.

Se você está aqui, provavelmente já ouviu falar sobre DevOps, sabe que ele se trata de **trabalhar de forma mais inteligente e eficiente**. E o Rancher está aí para ajudar nisso. Ele se conecta com ferramentas populares que as equipes de DevOps adoram, como Jenkins, Gitlab e Travis. Isso facilita a criação de etapas automatizadas para o desenvolvimento de seus projetos. Além disso, o Rancher oferece maneiras de acompanhar, registrar e manter seus aplicativos seguros.

Quando se trata de segurança, o Rancher sabe que isso é crucial. Ele permite que você ative a criptografia, também ajuda a configurar registros de auditoria, que funcionam como um registro de atividades, e verifica se suas configurações seguem as melhores práticas. Isso é importante para ter certeza de que tudo está seguro e em conformidade com os padrões estabelecidos.

Mas o Rancher vai além! Ele também oferece liberdade para inovar. Isso significa que você pode experimentar e criar sem medo. O melhor de tudo é que o Rancher é gratuito e de código aberto, o que significa que você não está preso a um único caminho. Se em algum momento você quiser mudar de direção, não terá problemas.

[1] Capacidade de monitorar, rastrear e compreender o comportamento e o desempenho de um sistema de maneira detalhada

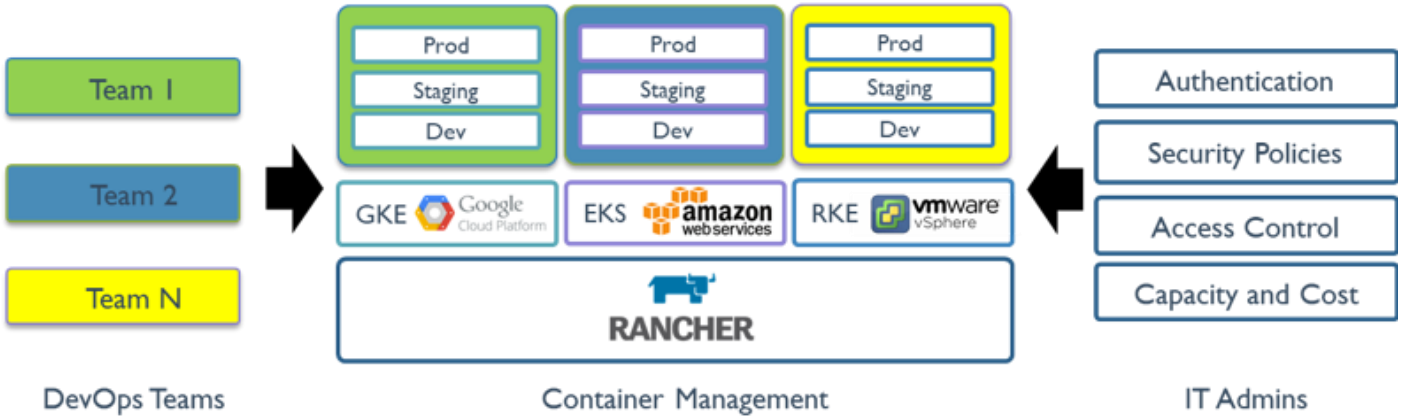
Overview

O Kubernetes se tornou a maneira comum de organizar e controlar contêineres. Muitas empresas de nuvem e virtualização agora oferecem o Kubernetes como parte de suas soluções. Usuários do Rancher têm diferentes formas de criar grupos de contêineres chamados de clusters. Eles podem usar o Rancher Kubernetes Engine (RKE) ou serviços de Kubernetes na nuvem, como GKE, AKS e EKS. Também podem trazer clusters Kubernetes já existentes de qualquer lugar e gerenciá-los pelo Rancher.

O Rancher também ajuda a atender às necessidades de gerenciamento de TI. Ele permite que você controle quem pode acessar esses clusters de maneira centralizada. Isso é útil, por exemplo, ao usar credenciais do Active Directory para entrar em clusters hospedados em nuvens como GKE. O Rancher também permite configurar regras de acesso e segurança para usuários, grupos, projetos, clusters e nuvens. Além disso, você pode ver o estado e a capacidade dos clusters em um só lugar.

O Rancher também é uma ferramenta amigável para as equipes de DevOps. Ele oferece uma interface de usuário fácil para que engenheiros de DevOps possam gerenciar os aplicativos que eles desenvolvem. Você não precisa ser um expert em Kubernetes para usar o Rancher. Rancher tem um catálogo de ferramentas úteis para DevOps e é compatível com várias ferramentas nativas em nuvem, como segurança, monitoramento, registro de contêineres e drivers de armazenamento e rede.

Na organização de TI e DevOps, o Rancher desempenha um papel importante. As diferentes equipes podem colocar seus aplicativos nas nuvens públicas ou privadas de sua escolha. Os administradores de TI podem ver tudo e aplicar regras para usuários, clusters e nuvens.



A plataforma do Rancher é construída com recursos que facilitam várias tarefas. O servidor de API do Rancher, baseado no Kubernetes, ajuda na autorização, controle de acesso e gerenciamento de usuários. Ele também lida com a criação e gerenciamento de clusters Kubernetes. Por exemplo, é possível criar projetos que agrupam várias partes de um cluster para melhor organização. Além disso, o Rancher ajuda na implantação contínua de aplicativos e na integração com ferramentas como o Istio para controle do tráfego de aplicativos.

Se você estiver trabalhando com infraestrutura em nuvem, o Rancher pode acompanhar e configurar nós automaticamente. Ele também ajuda a monitorar o estado dos seus clusters e pode se integrar com ferramentas populares de registro e monitoramento.

Por fim, o Rancher permite editar os clusters que você criou. Dependendo de como você os provisionou, as opções disponíveis podem variar. Depois de criados com o Rancher, os administradores podem gerenciar adesões, segurança e grupos de nós de maneira fácil.

Autenticação e Autorização

Um dos principais recursos que o Rancher adiciona ao Kubernetes é a autenticação centralizada do usuário. Este recurso permite configurar usuários locais e/ou conectar-se a um provedor de autenticação externo. Ao conectar-se a um provedor de autenticação externo, você pode aproveitar os usuários e grupos desse provedor.

No Rancher, cada pessoa se autentica como usuário, que é um login que concede acesso ao Rancher. Depois que o usuário faz login no Rancher, sua autorização ou direitos de acesso no sistema são determinados pela função do usuário. O Rancher fornece funções integradas para permitir que você configure facilmente as permissões de um usuário para recursos, mas o Rancher também oferece a capacidade de personalizar as funções para cada recurso do Kubernetes.

Políticas de segurança de pod (ou PSPs) são objetos que controlam aspectos sensíveis à segurança da especificação do pod, por exemplo, privilégios de root. Se um pod não atender às condições especificadas no PSP, o Kubernetes não permitirá que ele seja iniciado e o Rancher exibirá uma mensagem de erro.

Controle de acesso baseado em função (RBAC)

No Rancher, cada pessoa se autentica como usuário , que é um login que concede acesso ao Rancher. Conforme mencionado em Autenticação , os usuários podem ser locais ou externos.

Depois de configurar a autenticação externa, os usuários exibidos na página Usuários mudam.

- Se você estiver conectado como usuário local, somente usuários locais serão exibidos.
- Se você estiver conectado como usuário externo, os usuários externos e locais serão exibidos.

Provisionando Drivers

Os drivers no Rancher permitem que você gerencie quais provedores podem ser usados para implantar clusters ou nós hospedados do Kubernetes em um provedor de infraestrutura para permitir que o Rancher implante e gerencie o Kubernetes.

Com os drivers Rancher, você pode ativar/desativar os drivers integrados existentes que são empacotados no Rancher. Alternativamente, você pode adicionar seu próprio driver se o Rancher ainda não o tiver implementado.

Existem dois tipos de drivers no Rancher:

- Drivers de cluster
- Drivers de nó

Os **drivers de cluster** são usados para provisionar clusters Kubernetes hospedados, como GKE, EKS, AKS, etc. A disponibilidade de qual driver de cluster exibir ao criar um cluster é definida com base no status do driver de cluster. Somente activedrivers de cluster serão exibidos como uma opção para criar clusters para clusters Kubernetes hospedados. Por padrão, o Rancher é fornecido com vários drivers de cluster existentes, mas você também pode criar drivers de cluster personalizados para adicionar ao Rancher.

Os **drivers de nó** são usados para provisionar hosts, que o Rancher usa para iniciar e gerenciar clusters Kubernetes. Um driver de nó é igual a um driver de máquina Docker . A disponibilidade de qual driver de nó exibir ao criar modelos de nó é definida com base no status do driver de nó. Somente activedrivers de nó serão exibidos como uma opção para criação de modelos de nó. Por padrão, o Rancher é fornecido com muitos drivers Docker Machine existentes, mas você também pode criar drivers de nó personalizados para adicionar ao Rancher.

Políticas de segurança de pods

Políticas de segurança de pod (ou PSPs) são objetos que controlam aspectos sensíveis à segurança da especificação do pod (como privilégios de root). Se um pod não atender às condições especificadas no PSP, o Kubernetes não permitirá sua inicialização e o Rancher exibirá uma mensagem de erro de **Pod <NAME> is forbidden: unable to validate**

Os PSPs funcionam por meio de herança:

- Por padrão, os PSPs atribuídos a um cluster são herdados por seus projetos, bem como por quaisquer namespaces adicionados a esses projetos.
- Exceção: Namespaces que não são atribuídos a projetos não herdam PSPs, independentemente de o PSP estar atribuído a um cluster ou projeto. Como esses namespaces não têm PSPs, as implantações de carga de trabalho nesses namespaces falharão, que é o comportamento padrão do Kubernetes.

Você pode substituir o PSP padrão atribuindo um PSP diferente diretamente ao projeto. Quaisquer cargas de trabalho que já estejam em execução em um cluster ou projeto antes da atribuição de um PSP não serão verificadas se estiverem em conformidade com o PSP. As cargas de trabalho precisariam ser clonadas ou atualizadas para ver se passam no PSP.

Administração de Cluster

Depois de provisionar um cluster no Rancher, você poderá começar a usar recursos avançados do Kubernetes para implantar e dimensionar seus aplicativos conteinerizados em ambientes de desenvolvimento, teste ou produção.

Depois que os clusters forem provisionados no Rancher , os proprietários dos clusters precisarão gerenciá-los. Existem muitas opções diferentes de como gerenciar seu cluster. Por exemplo, usando o kubectl em conjunto com um arquivo kubeconfig para acessar um cluster, é possível gerenciar clusters Kubernetes de forma eficiente. O kubectl, uma ferramenta de linha de comando, tem a capacidade de acessar diversos tipos de clusters, inclusive aqueles implantados através do Rancher.

Além disso, o Rancher oferece uma série de funcionalidades para gerenciar diferentes aspectos de um cluster. A administração dos membros do cluster, que são os nós constituintes, pode ser realizada por meio

do Rancher UI, da API do Rancher ou ainda utilizando o kubectl. A edição e a atualização dos clusters também podem ser realizadas de várias maneiras, incluindo o uso do Rancher UI, da API do Rancher ou do kubectl. O mesmo se aplica à gestão dos próprios nós, que envolve tarefas como adicionar novos nós, remover nós e configurar suas definições.

No contexto de armazenamento, o Rancher oferece recursos para gerenciar volumes persistentes e classes de armazenamento. Esses volumes são fundamentais para manter dados mesmo após a remoção de nós do cluster, e essa gestão pode ser realizada por meio do Rancher UI, da API do Rancher ou do kubectl. A organização e administração de projetos, namespaces e cargas de trabalho para a gestão de aplicativos e serviços no cluster também é possível por meio do Rancher UI, da API do Rancher ou do kubectl. Essa estruturação facilita a gestão geral do ambiente. Os catálogos de aplicativos são ferramentas valiosas para encontrar e instalar aplicativos no cluster, e podem ser acessados via Rancher UI ou API do Rancher.

Além disso, o Rancher se destaca ao possibilitar a configuração de diversas ferramentas, como soluções de monitoramento, segurança e gerenciamento de dados. Tais configurações podem ser feitas por meio do Rancher UI ou da API do Rancher.

No âmbito de segurança, o Rancher oferece recursos para a execução de verificações de segurança nos clusters, auxiliando na identificação e correção de problemas. Entre outras funcionalidades, o Rancher também permite a alternância de certificados para manter a segurança, a realização de backups e restauração de clusters implantados por meio dele, e a limpeza de componentes do Kubernetes quando o Rancher não tem mais acesso aos clusters, mantendo a integridade e eficiência.

Por fim, o Rancher oferece a configuração de políticas de segurança de pod, como vimos anteriormente, que restringem o acesso aos pods, com opções de configuração tanto no Rancher UI quanto na API do Rancher.

(FCC- TRT 23 - 2022) Um Técnico precisa administrar e monitorar containers Docker em produção.

Para isso pode utilizar a ferramenta

a) Rancher
b) Webhooker
c) Swagger
d) Zuul
e) Flyway

Comentários: O Rancher é uma plataforma de código aberto projetada para administrar e monitorar ambientes de containers Docker, Kubernetes e outros em cenários de produção. Ele também permite implantar aplicativos usando containers Docker de maneira eficiente. O Rancher é uma ferramenta essencial no ecossistema DevOps, oferecendo recursos de gerenciamento de infraestrutura que abrangem desde o desenvolvimento até a produção.

Em relação ao Docker, o Rancher atua como um facilitador para o gerenciamento da infraestrutura Docker, tornando mais eficiente a administração e o monitoramento de containers em produção e também em ambientes de desenvolvimento.

No que diz respeito ao Kubernetes, o Rancher se destaca por oferecer uma interface de usuário mais intuitiva e amigável para a gestão de usuários e ambientes de containers. Isso simplifica o processo de gerenciamento em comparação com as abordagens mais complexas muitas vezes associadas ao Kubernetes. (**Gabarito**: Letra A)

Cluster Kubernetes K3s de Alta Disponibilidade

A escolha da infraestrutura recomendada para o cluster Kubernetes do Rancher varia dependendo de onde o Rancher será instalado: em um cluster Kubernetes K3s, um cluster Kubernetes RKE ou até mesmo em um único contêiner Docker.

Para instalar o servidor de gerenciamento do Rancher em um cluster K3s de alta disponibilidade, é recomendado configurar a seguinte infraestrutura:

- Dois nós Linux, normalmente máquinas virtuais, no provedor de infraestrutura que você escolher.
- Um banco de dados externo para armazenar os dados do cluster. A opção recomendada é o MySQL.
- Um balanceador de carga para distribuir o tráfego entre os dois nós.
- Um registro DNS para mapear uma URL para o balanceador de carga. Isso se tornará a URL do servidor Rancher, e os clusters Kubernetes secundários precisarão acessá-la.

Certifique-se de que seus nós atendam aos requisitos gerais de instalação em relação ao sistema operacional, runtime de contêineres, hardware e rede.

A infraestrutura recomendada para instalar o Rancher em um cluster Kubernetes K3 de alta disponibilidade é

a) dois nós Linux, normalmente máquina virtual, e um banco de dados externo do tipo NoSQL.
b) um cluster com Linux com menos dois nós ou quatro nós se Windows Server normalmente máquinas virtuais; com pelos menos 64 GB de memória RAM no provedor de infraestrutura e um banco de dados externo do tipo NoSQL.
c) um cluster com Linux de dois nós normalmente máquinas virtuais; com pelos menos 48 GB de memória RAM no provedor de infraestrutura.
d) dois nós Linux, normalmente máquinas virtuais; um banco de dados externo do tipo relacional, um balanceador de carga e um registro DNS.
e) quatro nós Windows Server, normalmente máquinas virtuais; um banco de dados externo do tipo relacional, um balanceador de carga entre os quatro nós e dois registros DNS como redundância.

Comentários: De acordo com a documentação oficial do Rancher, a infraestrutura recomendada para instalar o Rancher em um cluster Kubernetes K3 de alta disponibilidade é a seguinte:

Dois nós Linux, normalmente máquinas virtuais;
Um banco de dados externo do tipo relacional, como o MySQL ou o PostgreSQL;
Um balanceador de carga;
Um registro DNS.

O banco de dados externo é necessário para armazenar os dados do Rancher, como usuários, grupos, clusters e aplicativos. O balanceador de carga é necessário para distribuir o tráfego de entrada para os nós do

Rancher. O registro DNS é necessário para resolver os nomes de domínio dos nós do Rancher. **Gabarito**: Letra D

Configuração do Armazenamento de Dados Externo

A capacidade de executar o Kubernetes usando um armazenamento de dados diferente do etcd é o que diferencia o K3s de outras distribuições do Kubernetes. Isso oferece flexibilidade aos operadores do Kubernetes, permitindo a escolha do armazenamento que melhor atenda ao caso de uso específico. Para uma instalação de alta disponibilidade do K3s, é necessário configurar um banco de dados externo MySQL. O Rancher foi testado em clusters Kubernetes K3s usando o MySQL versão 5.7 como armazenamento de dados. Ao instalar o Kubernetes por meio do script de instalação do K3s, você fornece os detalhes necessários para que o K3s se conecte ao banco de dados.

Configuração do Balanceador de Carga

Quando usamos computadores para gerenciar muitos aplicativos e serviços, é importante ter uma maneira de garantir que, se algo der errado em um lugar, o outro continue funcionando para que tudo continue funcionando bem.

O balanceador de carga é como um supervisor que garante que o tráfego (como pessoas fazendo perguntas) seja distribuído corretamente para as pessoas disponíveis para responder (como servidores). Ele também ajuda a evitar que um problema em um dos servidores cause problemas para as pessoas que usam os serviços.

Ao configurar o Kubernetes, o K3s instala uma espécie de "recepcionista" chamado Traefik. Esse recepcionista escuta o tráfego que chega em um local especial (as portas 80 e 443) e direciona esse tráfego para os lugares certos (como redirecionar perguntas para as pessoas certas).

Quando você instala o Rancher, ele cria algo como uma instrução para esse recepcionista (o recurso de Ingress) para dizer a ele onde enviar o tráfego relacionado ao Rancher. Então, quando alguém tenta acessar o Rancher usando seu endereço (hostname), o recepcionista entende essa instrução e envia o tráfego para o local certo (os "pods" do Rancher em execução no cluster).

Assim, é necessário configurar um balanceador de carga para redirecionar o tráfego para a réplica do Rancher em ambos os nós. Isso evita que uma falha em um dos nós prejudique a comunicação com o servidor de gerenciamento Rancher. Quando o Kubernetes for configurado posteriormente, a ferramenta K3s implantará um controlador de Ingress Traefik. Esse controlador escutará nas portas 80 e 443 dos nós de trabalho, respondendo ao tráfego direcionado para hostnames específicos. Quando o Rancher for instalado (em um momento posterior), o sistema do Rancher criará um recurso de Ingress. Esse recurso instruirá o controlador de Ingress Traefik a ouvir o tráfego destinado ao hostname do Rancher. O controlador de Ingress Traefik, ao receber o tráfego direcionado ao hostname do Rancher, encaminhará esse tráfego para os pods do Rancher em execução no cluster. Conseguiu entender? 😉

Ao implementar essa configuração, é importante considerar se você deseja utilizar um balanceador de carga de Camada 4 ou Camada 7. Um balanceador de Camada 4 é mais simples e encaminha tráfego TCP para os nós. É recomendado configurar um balanceador de Camada 4 para redirecionar tráfego para as portas TCP/80 e TCP/443 dos nós no cluster de gerenciamento do Rancher. O controlador de Ingress no cluster

redirecionará o tráfego HTTP para HTTPS e fará o encerramento SSL/TLS na porta TCP/443. O tráfego direcionado à porta TCP/80 será encaminhado ao pod Ingress na implantação do Rancher.

Por outro lado, um balanceador de Camada 7 é mais complexo, porém oferece recursos adicionais. Ele é capaz de fazer o encerramento TLS no próprio balanceador de carga, em vez de no Rancher.
O encerramento TLS no balanceador de carga é um processo de segurança que acontece quando as informações viajam pela internet. Quando você envia dados de um lugar para outro na internet, é importante que esses dados estejam protegidos para que ninguém possa ver ou interferir neles durante o percurso.

TLS (Transport Layer Security) é como uma tecnologia de "trancar" esses dados em uma "caixa segura" virtual enquanto eles viajam pela internet. Essa caixa segura é chamada de "criptografia". É como se os dados fossem trancados em um cofre antes de serem enviados e só pudessem ser destrancados pelo destinatário correto.

Aqui entra o balanceador de carga. Ele é como um guardião na internet que ajuda a direcionar o tráfego e proteger as informações. Quando falamos de "encerramento TLS no balanceador de carga", significa que o próprio guardião (o balanceador de carga) está encarregado de destrancar as caixas seguras (a criptografia) dos dados quando eles chegam ao destino.

Isso pode ser útil para centralizar o encerramento TLS em sua infraestrutura. O balanceador de Camada 7 também permite que o balanceador tome decisões com base em atributos HTTP, como cookies, que um balanceador de Camada 4 não é capaz de considerar.

Caso opte por encerrar o tráfego SSL/TLS em um balanceador de Camada 7, é necessário usar a opção --set tls=external ao instalar o Rancher em um momento posterior.

Depois de configurar o seu balanceador de carga, você precisará criar um registro DNS para direcionar o tráfego para esse balanceador de carga. Dependendo do seu ambiente, isso pode ser um registro A que aponta para o endereço IP do balanceador de carga, ou pode ser um CNAME que aponta para o nome do host do balanceador de carga. Em ambos os casos, certifique-se de que esse registro seja o nome do host no qual você deseja que o Rancher responda.

Você precisará especificar esse nome do host em um passo posterior quando instalar o Rancher, e não será possível alterá-lo posteriormente. Portanto, certifique-se de que sua decisão seja definitiva.

QUESTÕES COMENTADAS

1. **(FGV – TRT - 13ª Região (PB) – 2022)** A infraestrutura recomendada para instalar o Rancher em um cluster Kubernetes K3 de alta disponibilidade é

a) dois nós Linux, normalmente máquina virtual, e um banco de dados externo do tipo NoSQL.
b) um cluster com Linux com menos dois nós ou quatro nós se Windows Server normalmente máquinas virtuais; com pelos menos 64 GB de memória RAM no provedor de infraestrutura e um banco de dados externo do tipo NoSQL.
c) um cluster com Linux de dois nós normalmente máquinas virtuais; com pelos menos 48 GB de memória RAM no provedor de infraestrutura.
d) dois nós Linux, normalmente máquinas virtuais; um banco de dados externo do tipo relacional, um balanceador de carga e um registro DNS.
e) quatro nós Windows Server, normalmente máquinas virtuais; um banco de dados externo do tipo relacional, um balanceador de carga entre os quatro nós e dois registros DNS como redundância.

Comentários:

De acordo com a documentação oficial do Rancher, a infraestrutura recomendada para instalar o Rancher em um cluster Kubernetes K3 de alta disponibilidade é a seguinte:

Dois nós Linux, normalmente máquinas virtuais;
Um banco de dados externo do tipo relacional, como o MySQL ou o PostgreSQL;
Um balanceador de carga;
Um registro DNS.

O banco de dados externo é necessário para armazenar os dados do Rancher, como usuários, grupos, clusters e aplicativos. O balanceador de carga é necessário para distribuir o tráfego de entrada para os nós do Rancher. O registro DNS é necessário para resolver os nomes de domínio dos nós do Rancher.

Gabarito: Letra D

---

2. **(FCC- TRT 23 - 2022)** Um Técnico precisa administrar e monitorar containers Docker em produção. Para isso pode utilizar a ferramenta

a) Rancher
b) Webhooker
c) Swagger
d) Zuul
e) Flyway

Comentários:

O Rancher é uma plataforma de código aberto projetada para administrar e monitorar ambientes de containers Docker, Kubernetes e outros em cenários de produção. Ele também permite implantar aplicativos usando containers Docker de maneira eficiente. O Rancher é uma ferramenta essencial no ecossistema

DevOps, oferecendo recursos de gerenciamento de infraestrutura que abrangem desde o desenvolvimento até a produção.

Em relação ao Docker, o Rancher atua como um facilitador para o gerenciamento da infraestrutura Docker, tornando mais eficiente a administração e o monitoramento de containers em produção e também em ambientes de desenvolvimento.

No que diz respeito ao Kubernetes, o Rancher se destaca por oferecer uma interface de usuário mais intuitiva e amigável para a gestão de usuários e ambientes de containers. Isso simplifica o processo de gerenciamento em comparação com as abordagens mais complexas muitas vezes associadas ao Kubernetes.

Gabarito: Letra A

LISTA DE QUESTÕES

1. **(FGV – TRT - 13ª Região (PB) – 2022)** A infraestrutura recomendada para instalar o Rancher em um cluster Kubernetes K3 de alta disponibilidade é

a) dois nós Linux, normalmente máquina virtual, e um banco de dados externo do tipo NoSQL.
b) um cluster com Linux com menos dois nós ou quatro nós se Windows Server normalmente máquinas virtuais; com pelos menos 64 GB de memória RAM no provedor de infraestrutura e um banco de dados externo do tipo NoSQL.
c) um cluster com Linux de dois nós normalmente máquinas virtuais; com pelos menos 48 GB de memória RAM no provedor de infraestrutura.
d) dois nós Linux, normalmente máquinas virtuais; um banco de dados externo do tipo relacional, um balanceador de carga e um registro DNS.
e) quatro nós Windows Server, normalmente máquinas virtuais; um banco de dados externo do tipo relacional, um balanceador de carga entre os quatro nós e dois registros DNS como redundância.

2. **(FCC- TRT 23 - 2022)** Um Técnico precisa administrar e monitorar containers Docker em produção. Para isso pode utilizar a ferramenta

a) Rancher
b) Webhooker
c) Swagger
d) Zuul
e) Flyway

GABARITO

1. Letra D
2. Letra A

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.