

## **Aula 01**

*TCE-PI (Auditor de Controle Externo -  
Área TI - Infraestrutura e Segurança)  
Passo Estratégico de Conhecimentos  
Específicos - 2024 (Pós-Edital)*

Autor:

**Fernando Pedrosa Lopes**

16 de Agosto de 2024

## DEVOPS

### Sumário

Conteúdo	1
Glossário de termos	3
Roteiro de revisão	6
Introdução	6
Conceitos e Princípios	8
Infraestrutura como Código (IAC)	10
Integração Contínua e Entrega Contínua (CI/CD)	13
Containers e Orquestração	16
Monitoramento e Logging	19
DevSecOps	21
<b>Aposta estratégica</b>	<b>23</b>
Questões Estratégicas	23
Questionário de revisão e aperfeiçoamento	27
Perguntas	28
Perguntas e Respostas	28
Lista de Questões Estratégicas	32
Gabaritos	34

## CONTEÚDO

Devops. Conceitos, práticas e princípios. Integração contínua. Entrega contínua. DevSecOps.



## ANÁLISE ESTATÍSTICA

Inicialmente, convém destacar o percentual de incidência do assunto, dentro da disciplina **Desenvolvimento de Sistemas** em concursos/cargos similares. Quanto maior o percentual de cobrança de um dado assunto, maior sua importância.

Obs.: *um mesmo assunto pode ser classificado em mais de um tópico devido à multidisciplinaridade de conteúdo.*

Assunto	Relevância na disciplina em concursos similares
Java SE, programação orientada a objetos e IDEs	16,04%
Estrutura de dados e algoritmos	11,68%
Web Services e REST	9,31%
Boas práticas de programação com testes automatizados, Clean Code, XP, FDD e TDD	9,31%
HTML e CSS	8,91%
Java Script, JQuery, Ajax, JSON e Spring	8,12%
Lógica de programação e sistemas de numeração	6,73%
PHP	5,94%
JDBC, JPA, Hibernate, JMS, EJB, JNDI e JTA	5,35%
<b>DEVOPS, DDD, Emergent Design, Refatoração, Docker, Kubernetes e Openshift</b>	<b>3,37%</b>
XML, XSLT e XSD	3,17%
Desenvolvimento mobile	2,57%
Compiladores e Interpretadores	2,38%
JavaEE/JakartaEE, JSP, JSF, JSTL, Servlets, Angular e React	2,18%



Controle de versão e monitoramento de código	2,18%
Arquitetura monolítica e microserviços	1,98%
Python e Ruby	< 1%
C, .NET, C# e ASP	< 1%

## GLOSSÁRIO DE TERMOS

*Faremos uma lista de termos que são relevantes ao entendimento do assunto desta aula. Caso tenha alguma dúvida durante a leitura, esta seção pode lhe ajudar a esclarecer.*

**DevOps:** Metodologia que une desenvolvimento e operações de TI para aumentar a velocidade, eficiência e qualidade na entrega de software.

**IAC (Infraestrutura como Código):** Prática de gerenciar e provisionar infraestruturas de TI usando scripts de código, permitindo automação e revisão de alterações.

**CI (Integração Contínua):** Prática de combinar todas as alterações de código de trabalho em um repositório centralizado continuamente, resultando em múltiplas integrações por dia.

**CD (Entrega Contínua):** Prática que visa a liberação rápida e confiável de software em produção, fazendo com que as alterações no código sejam seguras, rápidas e sustentáveis.

**Monitoramento:** Observação contínua e análise do desempenho do sistema para detectar e solucionar problemas ou falhas.

**Logging:** Prática de registrar eventos em um sistema, ajudando os desenvolvedores a entender o que aconteceu em caso de problemas.

**Terraform:** Ferramenta de Infraestrutura como Código para provisionar e gerenciar serviços na nuvem.

**Ansible:** Ferramenta de automação de TI para gerenciamento de configurações e orquestração de software.

**Puppet:** Ferramenta de gerenciamento de configurações que permite definir o estado desejado de sua infraestrutura e manter esse estado.



**CloudFormation:** Serviço da Amazon que ajuda a modelar e configurar recursos da Amazon Web Services.

**Jenkins:** Ferramenta de Integração Contínua de código aberto, usada para automação de partes do desenvolvimento de software.

**CircleCI:** Plataforma de Integração Contínua e Entrega Contínua, usada para automatizar o pipeline de DevOps.

**Travis CI:** Serviço de Integração Contínua usado para construir e testar projetos de software hospedados no GitHub e Bitbucket.

**GitLab CI/CD:** Ferramentas de Integração Contínua e Entrega Contínua incorporadas ao GitLab, uma plataforma de controle de versão baseada na web.

**Jenkins X:** Versão do Jenkins para desenvolvimento nativo de nuvem, focada em aplicações baseadas em Kubernetes.

**GitHub Actions:** Ferramenta do GitHub que permite a automação de fluxos de trabalho de software diretamente no repositório GitHub.

**Deployment Pipeline:** Processo de levar código do repositório para produção, normalmente composto por etapas de construção, teste e implantação.

**Container:** Unidade de software padronizada que inclui tudo o que o software precisa para ser executado.

**Docker:** Plataforma para desenvolver, empacotar e executar aplicações em containers.

**Dockerfile:** Arquivo de texto que contém os comandos para montar uma imagem Docker.

**Orquestração:** Prática de automatizar e coordenar tarefas complexas de computação e serviços.

**K8s (Kubernetes):** Plataforma de código aberto para automação de implantação, escalabilidade e gerenciamento de aplicações em containers.

**Manifesto Kubernetes:** Arquivo em formato YAML ou JSON que define os recursos necessários para executar uma aplicação ou serviço no Kubernetes.

**Prometheus:** Sistema de monitoramento e alerta de código aberto para métricas de tempo de série.



**Grafana:** Plataforma de visualização e análise de código aberto para métricas de tempo de série.

**Zabbix:** Software de monitoramento de rede de código aberto, usado para rastrear o status de vários serviços de rede.

**Datadog:** Serviço de monitoramento para nuvem que fornece monitoramento de servidores, bancos de dados, ferramentas e serviços.

**New Relic:** Plataforma de observabilidade que ajuda os desenvolvedores a rastrear, depurar e otimizar suas aplicações.

**ELK Stack (Elasticsearch, Logstash, Kibana):** Conjunto de três ferramentas de código aberto da Elastic para buscar, analisar e visualizar dados em tempo real.

**Grafana Loki:** Agregador de logs de código aberto que é integrado ao Grafana.

**Graylog:** Plataforma de gerenciamento de logs de código aberto para coleta, indexação e análise de logs.

**Splunk:** Software que é usado para pesquisar, monitorar e analisar dados de máquina gerados por operações de TI e segurança.

**DevSecOps:** Prática de integrar práticas de segurança em todas as fases do ciclo de desenvolvimento de software, combinando o desenvolvimento de software (Dev), segurança (Sec) e operações de TI (Ops).

## ROTEIRO DE REVISÃO

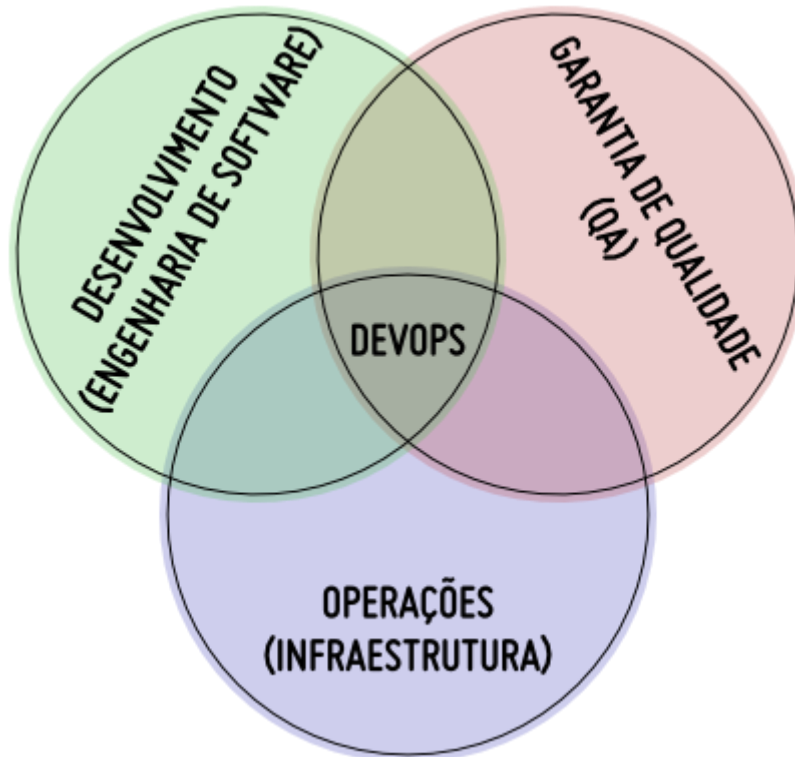
*A ideia desta seção é apresentar um roteiro para que você realize uma revisão completa do assunto e, ao mesmo tempo, destacar aspectos do conteúdo que merecem atenção.*

## Introdução

DevOps é um termo que combina as palavras "desenvolvimento" e "operações". **Refere-se a uma abordagem ou uma filosofia que promove uma melhor colaboração entre as equipes de desenvolvimento e de operações.** O objetivo é ajudar as organizações a



produzirem software e serviços mais rapidamente e com maior qualidade, melhorando e acelerando a comunicação e a colaboração entre estas duas equipes.



A ideia de DevOps começou a se formar em meados da década de 2000, quando as empresas começaram a adotar metodologias ágeis de desenvolvimento de software. No entanto, mesmo com a adoção dessas práticas ágeis, as equipes de operações de TI muitas vezes se viam lutando para acompanhar o ritmo rápido de desenvolvimento e implantação de novos recursos.

Em 2008, a frustração com a desconexão entre desenvolvimento e operações foi verbalizada no evento Agile Toronto. Andrew Clay Shafer e Patrick Debois propuseram uma palestra intitulada "Agile Infrastructure". Apesar do pouco interesse na época, a dupla continuou promovendo a ideia de operações ágeis.

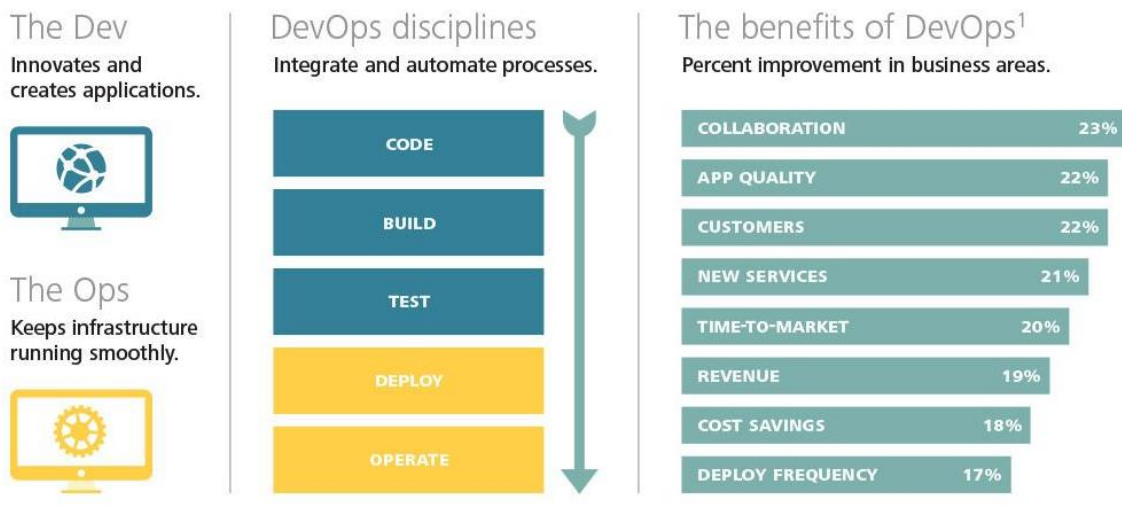
Em 2009, Paul Hammond e John Allspaw deram uma palestra no Velocity Conference intitulada "10+ Deploys Per Day: Dev and Ops Cooperation at Flickr", que se tornou um marco na jornada DevOps. Também em 2009, Debois organizou o primeiro evento DevOpsDays em Ghent, na Bélgica, e foi aí que o termo "DevOps" começou a ser amplamente adotado.

Desde então, a filosofia DevOps cresceu e se desenvolveu, e agora é adotada por muitas empresas ao redor do mundo.



Veja algumas das vantagens trazidas pela cultura DevOps:

- **Aumento da eficiência:** DevOps promove automação e monitoramento contínuos em todas as fases do ciclo de vida do software, desde a integração, teste, liberação até a implantação e gerenciamento de infraestrutura.
- **Rapidez na entrega:** Com a automação e a integração contínua, as equipes podem produzir e liberar software mais rapidamente e com mais frequência.
- **Melhoria na colaboração e comunicação:** Ao promover uma cultura de colaboração, DevOps ajuda a melhorar a comunicação e a colaboração entre as equipes de desenvolvimento e operações.
- **Qualidade do software:** DevOps permite às equipes capturarem e corrigir bugs mais cedo no ciclo de vida do desenvolvimento, o que leva a um produto final de maior qualidade.
- **Satisfação do cliente:** Com lançamentos mais rápidos, melhor qualidade de software e um tempo de resposta mais rápido para corrigir problemas, as organizações podem fornecer um melhor serviço ao cliente e aumentar a satisfação do cliente.



## Conceitos e Princípios

De forma geral, DevOps pode ser mais bem entendido ao olharmos mais de perto seus princípios, práticas e a cultura ao seu redor.

### Princípios





DevOps é fundamentado em uma série de princípios-chave que visam melhorar a colaboração entre as equipes de desenvolvimento e operações, aumentar a eficiência e acelerar a entrega de software. Alguns desses princípios são:

- **Infraestrutura como código (IAC):** Trata-se de gerir e provisionar a infraestrutura de TI através de código, permitindo que os desenvolvedores e operadores de sistemas automaticamente gerenciem e provisionem recursos de TI.
- **Integração contínua e entrega contínua (CI/CD):** A integração contínua envolve a automação do processo de compilação e teste do código sempre que uma alteração é feita. A entrega contínua estende isso para incluir a automação do processo de entrega.
- **Monitoramento e feedback contínuos:** Isso envolve monitorar constantemente o desempenho do aplicativo para detectar e corrigir problemas rapidamente.
- **Automação sempre que possível:** Ajuda a acelerar processos manuais, reduzir erros humanos e permitir que as equipes se concentrem em tarefas mais importantes.

### Práticas

Algumas das práticas comuns que as organizações seguem ao implementar DevOps incluem:

- **Adoção de uma mentalidade de "tudo como código":** Inclui infraestrutura, testes, segurança, conformidade e operações.
- **Criação de equipes multifuncionais:** Compreende desenvolvedores, engenheiros de operações, QA, engenheiros de segurança, e outros, todos trabalhando juntos em toda a duração de um projeto.
- **Implementação de CI/CD:** Isso permite que as alterações de código sejam integradas, testadas e implantadas de maneira mais rápida e eficiente.
- **Uso de contêineres e microsserviços:** Ajuda a criar arquiteturas mais escaláveis e flexíveis que podem ser desenvolvidas e implantadas de forma independente.
- **Implementação de monitoramento e observabilidade:** Permite que as equipes rastreiem a performance e a disponibilidade do sistema e respondam rapidamente a qualquer problema.

A cultura DevOps enfatiza a colaboração, a transparência e a responsabilidade compartilhada. As equipes são encorajadas a trabalhar juntas, compartilhar feedback e melhorar continuamente. A mentalidade de "nós contra eles" entre as equipes de desenvolvimento e operações é substituída pela mentalidade de "nós". Além disso, a falha é vista como uma oportunidade de aprendizado e a experimentação é encorajada.



### One team, one goal



## Infraestrutura como Código (IAC)

Infraestrutura como Código (IAC) é um **método de gerenciamento de infraestrutura de TI, em que tudo é gerenciado através de arquivos de código**. Isso inclui servidores, bancos de dados, redes, sistemas de armazenamento e outras infraestruturas.

Ao usar o IAC, a infraestrutura é provisionada e gerenciada automaticamente, sem a necessidade de configuração manual. Isso permite que as equipes de desenvolvimento e operações usem as mesmas ferramentas para gerenciar tanto a aplicação quanto a infraestrutura, o que facilita o controle de versão, o teste, a replicação de ambientes e muito mais.

Por exemplo, vamos supor que queremos criar e gerenciar infraestrutura na AWS (Amazon Web Services). Utilizando conceitos de IAC, os passos seriam, de forma geral, os seguintes:

### 1. Desenvolvimento de código de infraestrutura

O primeiro passo é escrever o código de infraestrutura. No caso do Terraform, isso é feito através de arquivos de configuração .tf. Esses arquivos descrevem a infraestrutura desejada. Por exemplo, o seguinte código define um provedor AWS e solicita a criação de uma instância EC2.



```
provider "aws" {  
  region = "us-west-2"  
}  
  
resource "aws_instance" "example" {  
  ami           = "ami-0c94855ba95c574c8"  
  instance_type = "t2.micro"  
  
  tags = {  
    Name = "example-instance"  
  }  
}
```

## 2. Planejamento de Infraestrutura

O próximo passo é a geração de um plano de execução com o comando **terraform plan**. Este comando compara o estado atual da infraestrutura com o estado desejado descrito no código, e gera um plano de ações necessárias para alcançar o estado desejado.

## 3. Avaliação de alterações propostas

A saída do comando **terraform plan** deve ser revisada para garantir que as ações propostas estejam corretas. Isto é especialmente importante porque algumas ações podem levar a perda de dados (por exemplo, a destruição de um banco de dados).

## 4. Aplicação de alterações de infraestrutura

Se o plano de execução for aceito, as alterações podem ser aplicadas com o comando **terraform apply**. Este comando realiza as ações necessárias para alcançar o estado desejado da infraestrutura.

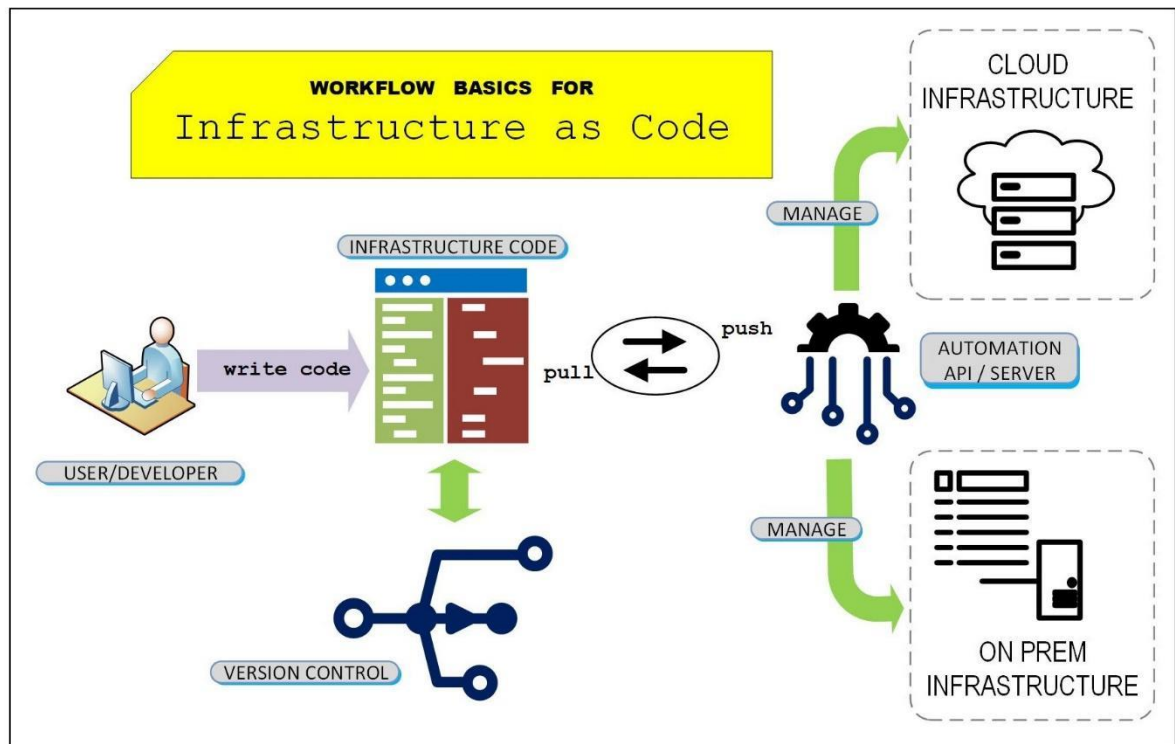
## 5. Monitoramento e manutenção da infraestrutura

Depois que a infraestrutura é criada, ela deve ser monitorada para garantir que está funcionando corretamente. Além disso, o código de infraestrutura pode precisar ser atualizado à medida que os requisitos mudam.

## 6. Desfazer mudanças, se necessário



Se em algum ponto for necessário reverter para um estado anterior da infraestrutura, o Terraform permite que isso seja feito com o comando **terraform destroy**.



### Ferramentas de IaC

Existem várias ferramentas de IAC, cada uma com seus próprios pontos fortes e fracos. Algumas das mais populares incluem:

- **Terraform**: uma ferramenta de IAC de código aberto que permite definir e provisionar infraestrutura de data center utilizando uma linguagem de descrição de alto nível.
- **Ansible**: uma ferramenta de automação de TI que pode configurar sistemas, implantar software e orquestrar tarefas mais complexas, como implantação contínua ou updates automáticos.
- **Puppet**: uma ferramenta que permite gerenciar a configuração de sistemas de forma declarativa.
- **CloudFormation (AWS)**: uma ferramenta da Amazon Web Services que ajuda a modelar e configurar toda a infraestrutura de TI para AWS.



## Integração Contínua e Entrega Contínua (CI/CD)

Duas práticas fundamentais na cultura DevOps são integração contínua e entrega contínua e suas respectivas ferramentas. Vamos estudar isso em mais detalhes.

### Integração Contínua (CI)

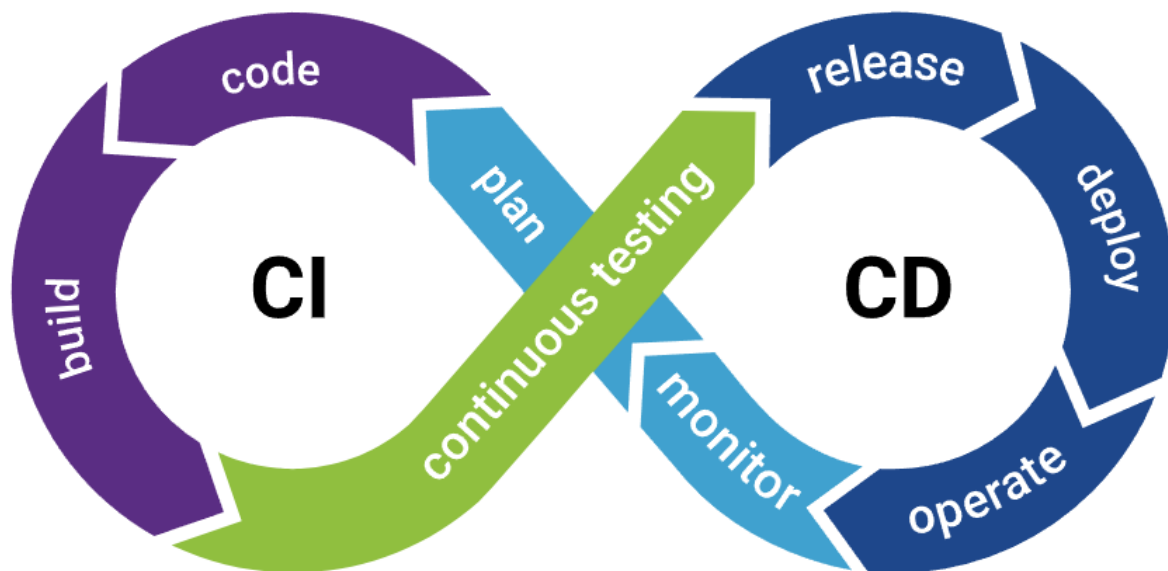
Integração Contínua (CI) é uma prática de desenvolvimento de software que envolve **integrar regularmente as alterações de código em um repositório central**. Após cada integração, as alterações de código são verificadas por meio de builds e testes automáticos. Isso permite detectar e corrigir problemas mais rapidamente, melhorar a qualidade do software e reduzir o tempo necessário para validar e lançar novas atualizações de software.

### Entrega Contínua (CD)

Entrega Contínua (CD) é a extensão natural da Integração Contínua: uma abordagem na qual as equipes de software garantem que cada alteração no código possa ser implantada em produção de forma segura e que possa ser lançada quando necessário. Isso é alcançado **garantindo que o código esteja sempre em um estado implantável** e automatizando todo o processo de lançamento.

**Há uma distinção importante a ser feita entre Entrega Contínua e Implantação Contínua.** Na Entrega Contínua, cada alteração que passa por todos os estágios do pipeline de produção está pronta para ser lançada para o cliente, mas o processo de lançamento ocorre manualmente. Em Implantação Contínua, este processo é automatizado, cada alteração que passa por todos os estágios do pipeline de produção é lançada automaticamente.





### Ferramentas CI/CD

Existem várias ferramentas que podem ajudar a implementar a Integração Contínua e a Entrega Contínua. As mais populares são:

- **Jenkins:** uma ferramenta de CI/CD de código aberto que suporta uma ampla gama de plugins e integrações. Jenkins é altamente configurável e pode ser adaptado para muitos diferentes pipelines de CI/CD.
- **CircleCI:** uma plataforma de CI/CD baseada em nuvem que suporta a execução de pipelines de CI/CD em várias linguagens de programação e frameworks.
- **Travis CI:** outra plataforma de CI/CD baseada em nuvem que é particularmente popular em projetos de código aberto.
- **GitLab CI/CD:** uma ferramenta de CI/CD que é integrada ao GitLab, uma plataforma de controle de versão baseada em Git.
- **Jenkins X:** uma versão do Jenkins projetada especificamente para Kubernetes e desenvolvimento de aplicativos baseados em nuvem.
- **GitHub Actions:** Uma ferramenta de automação de software que permite a desenvolvedores criar, testar e implantar aplicações diretamente do GitHub.

### Deployment Pipeline

Um pipeline de implantação (deployment pipeline) é um conceito chave nas práticas de entrega contínua (Continuous Delivery) e implantação contínua (Continuous Deployment).



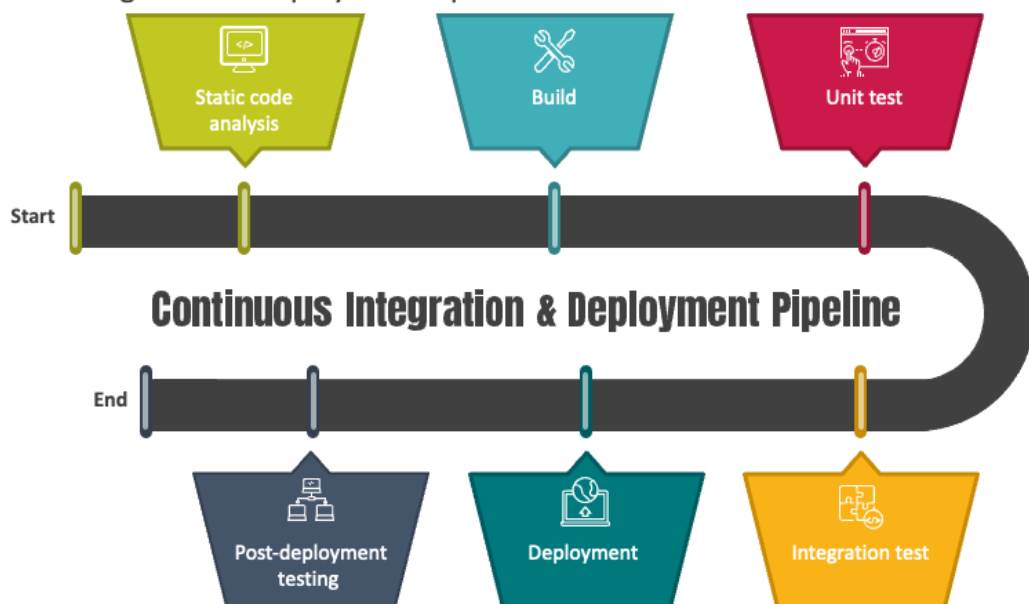
O pipeline é basicamente uma sequência de estágios pelos quais as alterações de código passam, desde a integração até a implantação em um ambiente de produção. Ele automatiza o processo de obtenção de código, realização de testes, e implantação de software.

Os estágios mais comuns de um deployment pipeline são:

1. **Integração Contínua:** Nesta etapa, as alterações de código são obtidas do sistema de controle de versão e compiladas. Testes unitários e outros testes rápidos são executados para validar as alterações.
2. **Teste de aceitação automatizado:** Aqui, o software é testado em um ambiente que imita o ambiente de produção o mais próximo possível. Isso pode incluir a execução de testes de aceitação automatizados para garantir que o software atenda aos critérios de aceitação.
3. **Implantação em ambiente de teste/staging:** Nesta etapa, o software é implantado em um ambiente de teste ou staging para mais testes. Inclui testes manuais ou testes exploratórios para descobrir quaisquer problemas que os testes automatizados possam ter perdido.
4. **Implantação em produção:** Finalmente, se o software passar por todos os estágios anteriores, ele é implantado em um ambiente de produção.

## DEPLOYMENT PIPELINE

Continuous Integration & Deployment Pipeline



O pipeline de implantação permite uma visibilidade clara do progresso das alterações de código à medida que passam por essas etapas, o que ajuda muito a identificar e corrigir problemas rapidamente.



## Containers e Orquestração

Container é uma tecnologia que **permite empacotar um aplicativo junto com todas as suas dependências em uma unidade padronizada para desenvolvimento de software**. Isso facilita a portabilidade entre ambientes de computação, uma vez que você pode ter certeza de que o aplicativo funcionará da mesma maneira, independentemente do local onde for executado.

Docker é uma das tecnologias de contêineres mais populares. Ele oferece uma maneira de automatizar a implantação, dimensionamento e operações de aplicativos em contêineres. O container Docker pode ser usado em várias tarefas, desde hospedar aplicativos da web até executar tarefas em segundo plano e microserviços.

### **Exemplo de como Docker pode ser utilizado:**

Vamos supor que temos um aplicativo da web simples escrito em Python e que usa Flask como um framework web e Redis como backend de armazenamento. Veja como você pode usar Docker para "contêinerizar" esse aplicativo:

Primeiro, você precisaria criar um arquivo **Dockerfile**, que especifica como criar a imagem do Docker. O arquivo é mais ou menos assim:





```
# Use a base image with Python installed
FROM python:3.8

# Install the Flask and Redis libraries
RUN pip install flask redis

# Copy the source code of the app into the container
COPY . /app

# Set the working directory
WORKDIR /app

# Set the command to run the app
CMD ["python", "app.py"]
```

Então você usaria o comando **docker build** para construir a imagem do Docker a partir do Dockerfile:

```
docker build -t my-flask-app .
```

Uma vez que a imagem está construída, você pode executar o aplicativo em um contêiner Docker usando o comando **docker run**:

Com isso, seu aplicativo estaria rodando em um contêiner Docker, e seria acessível no seu navegador em **localhost:5000**.

## Orquestração

A orquestração de contêineres é o processo de automação e gerenciamento de muitos aspectos envolvidos no trabalho com contêineres, incluindo a implantação, a rede e o escalonamento de contêineres. Isso facilita a atividade de gerenciar sistemas complexos que consistem em muitos contêineres que trabalham juntos.

**Kubernetes** (K8s) é uma das plataformas de orquestração de contêineres mais populares. Ele permite que você implante, escale e gerencie contêineres em clusters de servidores além



de executar em várias plataformas, desde servidores físicos em um data center até máquinas virtuais em provedores de nuvem pública.

### Exemplo de como Kubernetes pode ser utilizado:

Suponhamos que você tenha um aplicativo da web que está sendo executado em um contêiner Docker. Veja como você pode usar Kubernetes para gerenciar esse aplicativo:

Primeiro, você precisaria criar um arquivo de manifesto do Kubernetes, que descreve como o aplicativo deve ser executado. O arquivo seria algo assim:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-web-app
spec:
  containers:
  - name: my-web-app
    image: my-web-app:1.0.0
    ports:
    - containerPort: 80
```

Depois que o manifesto estiver pronto, você pode usar a ferramenta de linha de comando **kubectl** para aplicar o manifesto ao seu cluster Kubernetes:

```
kubectl apply -f my-web-app.yaml
```

Uma vez que o manifesto foi aplicado, Kubernetes irá garantir que um Pod esteja sempre executando com o seu aplicativo. Se o Pod falhar por qualquer motivo, Kubernetes irá automaticamente recriá-lo.



## Monitoramento e Logging

Em DevOps, a prática de monitoramento e logging é essencial para manter a saúde, a disponibilidade e o desempenho de um aplicativo ou infraestrutura. Vamos entender um pouco mais sobre cada uma dessas práticas.

### Monitoramento

Monitoramento é a prática de coletar e analisar métricas de um sistema para entender seu desempenho ao longo do tempo. Métricas comuns incluem utilização da CPU, uso de memória, largura de banda da rede, latência, número de solicitações por segundo e taxa de erros, entre outras.

Monitorar aplicações permite que as equipes de DevOps detectem problemas rapidamente antes que eles afetem os usuários. Por exemplo, se a utilização da CPU em um servidor está consistentemente alta, isso pode indicar que o servidor está sobrecarregado e pode precisar ser escalado.

Existem várias ferramentas de monitoramento disponíveis, como Prometheus, Grafana, Zabbix, Datadog, New Relic e muitas outras. A escolha da ferramenta certa depende de vários fatores, incluindo o tipo de sistema que você está monitorando, o volume de dados que você está coletando, e assim por diante.

A tabela a seguir mostra um resumo descritivo de cada uma dessas ferramentas:

Ferramenta	Descrição
Prometheus	<b>É um sistema de monitoramento e alerta de código aberto que coleta métricas de seus alvos em intervalos específicos, avalia regras de alerta, exibe métricas e pode acionar alertas se alguma condição for observada como verdadeira.</b>
Grafana	<b>Grafana é uma plataforma de visualização e análise de código aberto. Ele permite que você consulte, visualize, alerte e entenda suas métricas independentemente de onde estão armazenadas. É extensivamente utilizado para visualizar dados do Prometheus.</b>
Zabbix	<b>É uma solução de monitoramento de rede de código aberto. Ele é capaz de monitorar vários parâmetros de rede, de</b>



	<b>servidores e de aplicativos. Ele suporta a detecção de problemas e a visualização de dados.</b>
<b>Datadog</b>	<b>É uma plataforma de monitoramento e segurança de serviço completo para aplicações em nuvem. Ela integra e automatiza monitoramento de infraestrutura, aplicações, logs e mais em uma única plataforma.</b>
<b>New Relic</b>	<b>New Relic é uma plataforma de observabilidade em tempo real que ajuda a analisar, solucionar problemas e otimizar seus aplicativos de software. Ela oferece uma visão detalhada do desempenho do aplicativo de várias perspectivas, incluindo análise de infraestrutura, rastreamento de transações, monitoramento de aplicativos e muito mais.</b>

### Logging

Logging é a prática de coletar e armazenar logs, que são registros de eventos que ocorreram em um sistema. Os logs podem incluir mensagens de erro, registros de transações, registros de atividades do usuário e muito mais.

Logs acabam sendo uma fonte valiosa de informação quando se trata de entender o que está acontecendo dentro de um sistema. Por exemplo, se um usuário relatar um problema com um aplicativo, os logs podem ajudar a identificar a causa do problema.

Assim como com o monitoramento, existem várias ferramentas de logging disponíveis, como ELK Stack (Elasticsearch, Logstash, Kibana), Grafana Loki, Graylog e Splunk.

É importante notar que o monitoramento e o logging são complementares. O monitoramento pode ajudar a identificar que um problema está ocorrendo, enquanto os logs ajudam a entender por que o problema está ocorrendo. Juntos, eles fornecem uma visão abrangente do estado de um sistema. A tabela a seguir mostra um resumo descritivo de ferramentas de logging:

<b>Ferramenta</b>	<b>Descrição</b>
-------------------	------------------



ELK Stack (Elasticsearch, Logstash, Kibana)	ELK Stack é uma combinação de três ferramentas de código aberto da Elastic: Elasticsearch, uma ferramenta de pesquisa e análise; Logstash, um servidor de coleta, processamento e encaminhamento de logs; e Kibana, uma ferramenta de visualização de dados. Juntas, elas fornecem uma solução de gerenciamento de logs de ponta a ponta.
Grafana Loki	Loki é uma ferramenta de agregação de logs de código aberto, criada pela Grafana Labs. Ela foi projetada para ser integrada facilmente com a plataforma de visualização Grafana. Loki não indexa o conteúdo dos logs, apenas os metadados, o que a torna uma ferramenta de log mais econômica e eficiente.
Graylog	Graylog é uma plataforma de gerenciamento de logs de código aberto que suporta coleta de logs e análise em tempo real. Ela inclui recursos como uma interface de usuário web para pesquisar conteúdo de log, dashboards, alertas e muito mais.
Splunk	Splunk é uma plataforma de software para buscar, monitorar e analisar dados de máquinas gerados por websites, aplicativos, servidores e dispositivos de rede móveis. Ela fornece funcionalidades para coleta e armazenamento de dados, criação de dashboards, alertas, visualizações e muito mais. Splunk é particularmente conhecida por sua poderosa capacidade de busca em logs.

## DevSecOps

DevSecOps é uma filosofia ou prática cultural que tem como objetivo a **integração da segurança em todas as fases do ciclo de vida do desenvolvimento de software, desde o planejamento até a manutenção**. O termo é uma junção de "Desenvolvimento", "Segurança" e "Operações".

O objetivo do DevSecOps é tornar todos dentro de uma organização responsáveis pela segurança, incorporando a segurança desde o início do ciclo de vida de desenvolvimento de



aplicativos, e não a tratar como uma fase posterior. Isso é feito para minimizar a ocorrência de vulnerabilidades de segurança no software final, melhorando a qualidade geral e a confiabilidade do software.

### Princípios do DevSecOps

- **Segurança como código:** A segurança deve ser tratada da mesma forma que o código, o que significa que ela deve ser gerenciada por meio de ferramentas de controle de versão, testada automaticamente e refatorada conforme necessário.
- **Tudo como Código (Infrastructure as Code, Policy as Code, Security as Code):** A ideia aqui é que a infraestrutura, política e segurança do aplicativo podem ser definidas e gerenciadas como código. Isso permite a automação, a repetibilidade e a consistência.
- **Colaboração e compartilhamento:** Como todos na organização são responsáveis pela segurança, é importante promover a colaboração e o compartilhamento de conhecimento entre desenvolvimento, operações e segurança.
- **Automação:** As verificações de segurança devem ser automatizadas tanto quanto possível. Isso pode incluir a automação de testes de segurança, varreduras de vulnerabilidade, análise estática de código e muito mais.
- **Monitoramento e resposta contínua:** O monitoramento de eventos de segurança e a resposta a incidentes devem ser feitos continuamente. Isso permite a detecção precoce e a correção de problemas de segurança.
- **Cultura de segurança:** A segurança deve ser uma parte central da cultura da empresa. Isso pode envolver treinamento regular, sessões de conscientização sobre segurança e a promoção de práticas seguras de codificação.

### Ferramentas de DevSecOps

Existem muitas ferramentas que podem ser usadas para apoiar práticas de DevSecOps. Alguns exemplos incluem:

- Ferramentas de análise de código estático (SAST), como SonarQube e Fortify, que podem detectar vulnerabilidades no código.
- Ferramentas de análise de código dinâmico (DAST), como OWASP ZAP e Nessus, que podem detectar vulnerabilidades em aplicativos em execução.
- Ferramentas de gerenciamento de vulnerabilidades, como OpenVAS e Nexpose, que podem ajudar a gerenciar e rastrear vulnerabilidades.
- Ferramentas de verificação de dependência, como OWASP Dependency Check e Snyk, que podem verificar as bibliotecas e dependências do projeto em busca de vulnerabilidades conhecidas.
- Ferramentas de automação de segurança de infraestrutura, como Ansible, Chef, Puppet e Terraform, que podem garantir que a infraestrutura seja configurada de maneira segura.



## APOSTA ESTRATÉGICA

*A ideia desta seção é apresentar os pontos do conteúdo que mais possuem chances de serem cobrados em prova, considerando o histórico de questões da banca em provas de nível semelhante à nossa, bem como as inovações no conteúdo, na legislação e nos entendimentos doutrinários e jurisprudenciais<sup>1</sup>.*

Infraestrutura como Código (IaC) é uma prática que transformou significativamente a maneira como as organizações gerenciam e provisionam infraestruturas de TI. Utilizando código para automatizar a configuração e o gerenciamento de hardware e software, o IaC elimina a necessidade de configurações manuais que são propensas a erros, reduzindo inconsistências entre os ambientes de desenvolvimento, teste e produção. Esta abordagem permite que os administradores de sistemas e desenvolvedores usem técnicas de controle de versão, como as aplicadas no desenvolvimento de software, para manter e rastrear mudanças na infraestrutura, melhorando a eficiência operacional e a resposta às mudanças no mercado.

DevSecOps integra práticas de segurança em todas as fases do desenvolvimento de software e operações, enfatizando a importância de incluir a segurança como um componente integrado, e não como um acréscimo tardio no ciclo de desenvolvimento. Ao adotar esta prática, as organizações podem identificar e corrigir vulnerabilidades de segurança mais rapidamente, o que reduz o risco de comprometimento de dados e melhora a conformidade regulatória. O DevSecOps promove uma cultura onde a segurança é uma responsabilidade compartilhada entre todos que estão envolvidos no projeto, incentivando a colaboração e a comunicação contínuas, e garantindo que as medidas de segurança acompanhem o ritmo acelerado das práticas de desenvolvimento e operações.

## QUESTÕES ESTRATÉGICAS

*Nesta seção, apresentamos e comentamos uma amostra de questões objetivas selecionadas estrategicamente: são questões com nível de dificuldade semelhante ao que você deve esperar para a sua prova e que, em conjunto, abordam os principais pontos do assunto.*

---

<sup>1</sup> Vale deixar claro que nem sempre será possível realizar uma aposta estratégica para um determinado assunto, considerando que às vezes não é viável identificar os pontos mais prováveis de serem cobrados a partir de critérios objetivos ou minimamente razoáveis.



*A ideia, aqui, não é que você fixe o conteúdo por meio de uma bateria extensa de questões, mas que você faça uma boa revisão global do assunto a partir de, relativamente, poucas questões.*

**1. (FCC/2018/Auditor Fiscal da Receita Estadual (SEF SC)/Tecnologia da Informação) No âmbito da infraestrutura, uma das vantagens DevOps é**

- a) realizar deploy sob controle manual de especialistas.
- b) estabelecer uma divisão bem delimitada entre infraestrutura e desenvolvimento.
- c) possuir um ambiente aberto (não padrão) sem muitos controles.
- d) ter a infraestrutura como código.
- e) fazer com que cada etapa do processo seja aprovada formalmente pelos gerentes.

**Comentários**

**Letra D – Correta.** Ter a infraestrutura como código é uma das características do DevOps. Isso permite entregar uma infraestrutura de forma rápida e automática.

**Letra A – Incorreta.** O deploy é feito de forma automática.

**Letra B – Incorreta.** O DevOps preconiza uma maior integração entre as áreas de desenvolvimento e infraestrutura.

**Letra C – Incorreta.** O ambiente contém controles, testes automatizados, etc.

**Letra E – Incorreta.** Cada etapa do processo não precisa ser formalmente aprovada pelos gerentes, até por que os processos são automatizados.

**Gabarito: D**

**2. (CESPE / TRT-PA e AP – 2016) Acerca de DevOps, assinale a opção correta.**

- a) O DevOps concentra-se em reunir diferentes processos e executá-los mais rapidamente e com mais frequência, o que gera baixa colaboração entre equipes.
- b) O DevOps tem como princípio produzir, a partir da avaliação dos times de desenvolvimento do serviço, grandes mudanças e farta documentação com valor agregado para os usuários, assemelhando-se, por isso, com objetivos dos métodos iterativos e em cascata.





c) A infraestrutura de nuvem de provedores internos e externos vem restringindo o uso de DevOps pelas organizações.

d) O DevOps parte da premissa de adoção de grandes equipes de especialistas, com a menor interação possível, visando à padronização de processos e à mínima automação de atividades.

e) Atividades típicas em DevOps compreendem teste do código automatizado, automação de fluxos de trabalho e da infraestrutura e requerem ambientes de desenvolvimento e produção idênticos.

**Comentários:** (a) Errado, isso gera alta colaboração entre equipes; (b) Errado, não se assemelha com objetivos de métodos em cascata, visto que essa metodologia possui entregas menos frequentes e é menos dinâmica; (c) Errado, é justamente o inverso – elas usam com cada vez mais frequência; (d) Errado, recomenda-se a maior interação possível entre as equipes; (e) Correto, testes automatizados, automação de fluxos e infraestrutura são atividades frequentes que realmente exigem ambientes de desenvolvimento e produção idênticos (Letra E).

**Gabarito: E**

### 3. (CESPE / TRE-PE – 2017) O DevOps consiste em:

a) um processo similar ao IRUP (IBM Rational Unified Process), que tem como objetivo dividir o processamento em fases e disciplinas de software para paralelizar as ações de desenvolvimento e de manutenção das soluções.

b) uma plataforma aberta cuja função é substituir a virtualização de aplicações e serviços em containers e, com isso, agilizar a implantação de soluções de software.

c) um aplicativo que permite o gerenciamento de versões de códigos-fonte e versões de programas, bem como a implantação da versão mais recente de um software em caso de falha.

d) um processo de promoção de métodos que objetivam aprimorar a comunicação, tornando a colaboração eficaz especialmente entre os departamentos de desenvolvimento e teste e entre os departamentos de operações e serviço para o negócio.

e) uma metodologia ágil que, assim como a XP (extreme programming) e o Scrum, tem foco na gestão de produtos complexos relativos à equipe de desenvolvimento.

**Comentários:** DevOps não é um processo similar ao iRUP, não é uma plataforma aberta, não é um aplicativo e não é uma metodologia ágil. DevOps é um processo de promoção de



métodos que objetivam aprimorar a comunicação, tornando a colaboração eficaz especialmente entre os departamentos de desenvolvimento e teste e entre os departamentos de operações e serviço para o negócio (Letra D).

**Gabarito: D**

**4. (CESPE / CEBRASPE - 2022 - TRT - 8ª Região (PA e AP) - Analista Judiciário - Tecnologia da Informação) 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:**

A alternativa D é correta porque define corretamente o conceito de Desenvolvimento Orientado a Testes (TDD, Test Driven Development). No TDD, os testes são escritos antes do código de produção. Este é um princípio fundamental do TDD e uma grande mudança em relação à prática tradicional de escrever testes após o código.

O processo de TDD geralmente segue estes passos:

- Escrever um teste para a nova funcionalidade que será implementada. Nesse ponto, o teste irá falhar porque a funcionalidade ainda não foi implementada.
- Escrever apenas o código suficiente para fazer o teste passar.
- Refatorar o código para garantir que ele esteja limpo e claro.
- Repetir o processo para a próxima funcionalidade.

**Gabarito: D**

**5. (CESGRANRIO - 2021 - Caixa - Técnico Bancário Novo - Tecnologia da Informação) No âmbito de DevOps, o termo “shift left testing” significa que os testes devem**

A) ser feitos junto com a entrada em operação do produto.



- B) ser feitos apenas no ambiente de desenvolvimento.
- C) ser feitos apenas por meio de ferramentas de automação de testes.
- D) ser feitos desde as fases iniciais do ciclo de vida do produto.
- E) dirigir o desenvolvimento do produto.

### Comentários:

O termo "shift left testing" se refere à prática de realizar testes o mais cedo possível no ciclo de desenvolvimento de software. Em vez de esperar até o final do ciclo de desenvolvimento para começar a testar, como era comum em muitos métodos tradicionais de desenvolvimento, a ideia é "deslocar" esses testes para a "esquerda" - ou seja, para as etapas anteriores.

A razão para isso é que os problemas costumam ser mais fáceis (e mais baratos) de corrigir se forem detectados cedo. Isso pode ajudar a melhorar a qualidade do software e reduzir o tempo geral de desenvolvimento. Também se alinha bem com as práticas de DevOps e Agile, que enfatizam a entrega contínua e a integração contínua.

### Gabarito: D

## QUESTIONÁRIO DE REVISÃO E APERFEIÇOAMENTO

*A ideia do questionário é elevar o nível da sua compreensão no assunto e, ao mesmo tempo, proporcionar uma outra forma de revisão de pontos importantes do conteúdo, a partir de perguntas que exigem respostas subjetivas.*

*São questões um pouco mais desafiadoras, porque a redação de seu enunciado não ajuda na sua resolução, como ocorre nas clássicas questões objetivas.*

*O objetivo é que você realize uma auto explicação mental de alguns pontos do conteúdo, para consolidar melhor o que aprendeu ;)*

*Além disso, as questões objetivas, em regra, abordam pontos isolados de um dado assunto. Assim, ao resolver várias questões objetivas, o candidato acaba memorizando pontos isolados do conteúdo, mas muitas vezes acaba não entendendo como esses pontos se conectam.*

*Assim, no questionário, buscaremos trazer também situações que ajudem você a conectar melhor os diversos pontos do conteúdo, na medida do possível.*



*É importante frisar que não estamos adentrando em um nível de profundidade maior que o exigido na sua prova, mas apenas permitindo que você compreenda melhor o assunto de modo a facilitar a resolução de questões objetivas típicas de concursos, ok?*

*Nosso compromisso é proporcionar a você uma revisão de alto nível!*

*Vamos ao nosso questionário:*

## Perguntas

1. O que é DevOps e por que é importante?
2. O que é Infraestrutura como Código (IAC) e por que é importante?
3. O que é Integração Contínua (CI) e por que é importante?
4. O que é Entrega Contínua (CD) e por que é importante?
5. O que é um Container e como o Docker o utiliza?
6. Como o Kubernetes ajuda na orquestração de containers?
7. O que é um pipeline de implantação e por que é importante?
8. O que é monitoramento em DevOps e por que é importante?
9. O que é logging em DevOps e por que é importante?
10. O que é o Terraform e para que é usado?
11. O que é o Ansible e para que é usado?
12. Como o Jenkins auxilia na Integração Contínua?
13. Como o GitLab CI/CD é usado em um pipeline de implantação?
14. O que é o Dockerfile e para que é usado?
15. O que é o Manifesto Kubernetes e para que é usado?
16. O que é Prometheus e para que é usado?
17. O que é DevSecOps e por que é importante?
18. Como o Zabbix é usado para o monitoramento de redes?
19. Quais são algumas práticas comuns em DevSecOps?
20. Como a prática de "Shift Left" se aplica em DevSecOps?

## Perguntas e Respostas

1. O que é DevOps e por que é importante?

Resposta: DevOps é uma metodologia que combina práticas de desenvolvimento de



software e operações de TI para aumentar a velocidade, eficiência e qualidade na entrega de software. É importante porque ajuda a reduzir o tempo de entrega, aumentar a frequência de implantação, diminuir a taxa de falhas de novas versões e melhorar o tempo de recuperação.

2. O que é Infraestrutura como Código (IAC) e por que é importante?

Resposta: Infraestrutura como Código (IAC) é a prática de gerenciar e provisionar infraestruturas de TI usando scripts de código. Isso permite automação, revisão de alterações e reutilização de configurações, tornando a infraestrutura mais confiável e eficiente.

3. O que é Integração Contínua (CI) e por que é importante?

Resposta: Integração Contínua (CI) é a prática de combinar todas as alterações de código em um repositório centralizado de maneira contínua. Isso promove a detecção precoce de problemas, evitando grandes problemas de integração no final do ciclo de desenvolvimento.

4. O que é Entrega Contínua (CD) e por que é importante?

Resposta: A Entrega Contínua (CD) é a prática de entregar software em pequenas e frequentes iterações, de forma que possa ser liberado para produção a qualquer momento. Isso aumenta a velocidade de entrega e a capacidade de resposta às mudanças do mercado.

5. O que é um Container e como o Docker o utiliza?

Resposta: Um Container é uma unidade padrão de software que empacota o código e todas as suas dependências para que o aplicativo seja executado de forma rápida e confiável de um ambiente de computação para outro. O Docker é uma plataforma que usa a tecnologia de containerização para empacotar e executar aplicativos.

6. Como o Kubernetes ajuda na orquestração de containers?

Resposta: Kubernetes é uma plataforma de código aberto para automação de implantação, escalabilidade e gerenciamento de aplicações em containers. Ele agrupa containers que compõem uma aplicação em unidades lógicas para fácil gerenciamento e descoberta de serviços.

7. O que é um pipeline de implantação e por que é importante?

Resposta: Um pipeline de implantação é um conjunto de processos que levam o código do repositório para o ambiente de produção. É importante para automação e consistência, permitindo entregas rápidas, confiáveis e repetíveis.

8. O que é monitoramento em DevOps e por que é importante?

Resposta: O monitoramento em DevOps é a prática de rastrear e analisar o desempenho do sistema e dos aplicativos ao longo do tempo. É crucial para identificar e resolver problemas antes que eles afetem os usuários finais ou a eficiência dos negócios.

9. O que é logging em DevOps e por que é importante?

Resposta: Logging é a prática de registrar eventos em um sistema. É importante para o diagnóstico de problemas, a compreensão do comportamento do sistema e a identificação de tendências ou insights para tomada de decisões.



10. O que é o Terraform e para que é usado?

Resposta: Terraform é uma ferramenta de Infraestrutura como Código para provisionar e gerenciar serviços na nuvem. É usado para criar, alterar e melhorar a infraestrutura de maneira segura e eficiente.

11. O que é o Ansible e para que é usado?

Resposta: Ansible é uma ferramenta de automação de TI para gerenciamento de configurações e orquestração de software. É usado para automatizar tarefas como provisionamento de servidores, implantação de aplicativos e orquestração de fluxos de trabalho complexos.

12. Como o Jenkins auxilia na Integração Contínua?

Resposta: Jenkins é uma ferramenta de Integração Contínua que automatiza várias fases do ciclo de desenvolvimento de software, como a construção, teste e implantação do código, permitindo a detecção precoce de problemas e a entrega contínua de software de qualidade.

13. Como o GitLab CI/CD é usado em um pipeline de implantação?

Resposta: GitLab CI/CD é uma ferramenta de Integração Contínua e Entrega Contínua incorporada ao GitLab. Ele pode ser usado para automatizar várias etapas do pipeline de implantação, como compilação, teste, empacotamento e implantação do código.

14. O que é o Dockerfile e para que é usado?

Resposta: Dockerfile é um arquivo de texto que contém os comandos para construir uma imagem Docker. Ele define os passos necessários para criar o ambiente de execução para um aplicativo, incluindo a base do sistema operacional, dependências do aplicativo e configurações.

15. O que é o Manifesto Kubernetes e para que é usado?

Resposta: Um Manifesto Kubernetes é um arquivo em formato YAML ou JSON que define os recursos necessários para executar uma aplicação ou serviço no Kubernetes. É usado para declarar contêineres, redes e volumes, entre outros componentes do Kubernetes.

16. O que é Prometheus e para que é usado?

Resposta: Prometheus é um sistema de monitoramento e alerta de código aberto para métricas de tempo de série. É usado para coletar e armazenar

17. Como a pilha ELK (Elasticsearch, Logstash, Kibana) é usada para gerenciar logs?  
Resposta: A pilha ELK é usada para coletar, armazenar, indexar e visualizar dados de log. Logstash coleta e transforma os dados, Elasticsearch os armazena e indexa, e Kibana os visualiza.

17. O que é DevSecOps e por que é importante?

Resposta: DevSecOps é a prática de integrar práticas de segurança em todas as fases do ciclo de desenvolvimento de software. É importante porque ajuda a identificar e corrigir vulnerabilidades de segurança mais cedo, reduzindo o risco e aprimorando a confiabilidade do software.

18. Como o Zabbix é usado para o monitoramento de redes?

Resposta: Zabbix é um software de monitoramento de rede de código aberto, usado para



rastrear o status de vários serviços de rede, servidores e outros dispositivos de rede. Ele coleta e visualiza dados de desempenho para permitir a detecção rápida de problemas.

19. Quais são algumas práticas comuns em DevSecOps?

Resposta: Algumas práticas comuns em DevSecOps incluem: análise de código estática para identificar vulnerabilidades de segurança no código-fonte, testes de segurança automatizados como parte do pipeline de CI/CD, uso de ferramentas de gerenciamento de vulnerabilidades para rastrear e gerenciar vulnerabilidades conhecidas, treinamento de desenvolvedores em práticas seguras de codificação, e a implementação de políticas de mínimos privilégios e controle de acesso baseado em função.

20. Como a prática de "Shift Left" se aplica em DevSecOps?

Resposta: "Shift Left" em DevSecOps refere-se à prática de integrar a segurança no início do ciclo de desenvolvimento de software, ao invés de tratar a segurança como uma consideração posterior. Isso pode envolver a inclusão de profissionais de segurança na equipe de desenvolvimento, a realização de revisões de segurança do código e design, e a automação de testes de segurança como parte do pipeline de CI/CD. A ideia é identificar e corrigir problemas de segurança mais cedo, quando eles são mais fáceis e menos custosos para resolver.

## LISTA DE QUESTÕES ESTRATÉGICAS

1. **(FCC/2018/Auditor Fiscal da Receita Estadual (SEF SC)/Tecnologia da Informação)** No âmbito da infraestrutura, uma das vantagens DevOps é
  - a) realizar deploy sob controle manual de especialistas.
  - b) estabelecer uma divisão bem delimitada entre infraestrutura e desenvolvimento.
  - c) possuir um ambiente aberto (não padrão) sem muitos controles.
  - d) ter a infraestrutura como código.
  - e) fazer com que cada etapa do processo seja aprovada formalmente pelos gerentes.
2. **(CESPE / TRT-PA e AP – 2016)** Acerca de DevOps, assinale a opção correta.
  - a) O DevOps concentra-se em reunir diferentes processos e executá-los mais rapidamente e com mais frequência, o que gera baixa colaboração entre equipes.



- b) O DevOps tem como princípio produzir, a partir da avaliação dos times de desenvolvimento do serviço, grandes mudanças e farta documentação com valor agregado para os usuários, assemelhando-se, por isso, com objetivos dos métodos iterativos e em cascata.
- c) A infraestrutura de nuvem de provedores internos e externos vem restringindo o uso de DevOps pelas organizações.
- d) O DevOps parte da premissa de adoção de grandes equipes de especialistas, com a menor interação possível, visando à padronização de processos e à mínima automação de atividades.
- e) Atividades típicas em DevOps compreendem teste do código automatizado, automação de fluxos de trabalho e da infraestrutura e requerem ambientes de desenvolvimento e produção idênticos.

**3. (CESPE / TRE-PE – 2017) O DevOps consiste em:**

- a) um processo similar ao IRUP (IBM Rational Unified Process), que tem como objetivo dividir o processamento em fases e disciplinas de software para paralelizar as ações de desenvolvimento e de manutenção das soluções.
- b) uma plataforma aberta cuja função é substituir a virtualização de aplicações e serviços em containers e, com isso, agilizar a implantação de soluções de software.
- c) um aplicativo que permite o gerenciamento de versões de códigos-fonte e versões de programas, bem como a implantação da versão mais recente de um software em caso de falha.
- d) um processo de promoção de métodos que objetivam aprimorar a comunicação, tornando a colaboração eficaz especialmente entre os departamentos de desenvolvimento e teste e entre os departamentos de operações e serviço para o negócio.
- e) uma metodologia ágil que, assim como a XP (extreme programming) e o Scrum, tem foco na gestão de produtos complexos relativos à equipe de desenvolvimento.

**4. (CESPE / CEBRASPE - 2022 - TRT - 8ª Região (PA e AP) - Analista Judiciário - Tecnologia da Informação) 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.

5. **(CESGRANRIO - 2021 - Caixa - Técnico Bancário Novo - Tecnologia da Informação)** No âmbito de DevOps, o termo “shift left testing” significa que os testes devem

A) ser feitos junto com a entrada em operação do produto.

B) ser feitos apenas no ambiente de desenvolvimento.

C) ser feitos apenas por meio de ferramentas de automação de testes.

D) ser feitos desde as fases iniciais do ciclo de vida do produto.

E) dirigir o desenvolvimento do produto.

## Gabaritos

1. D
2. E
3. D
4. D
5. D



# 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.