BOOTCAMP DEVOPS CLOUD

CI/CD

O que é CI/CD?

Integração Contínua (CI) e Entrega/Implantação Contínua (CD) são formas de entregar aplicativos aos clientes, empacotando o código, criando artefatos e implementando em um sistema.

- CI/CD introduz Automação e Monitoramento contínuo (Lifecycle);
- Possui fases como Integração, Teste, EntregA e Implantação;
- Essas práticas são frequentemente chamadas de "Pipeline CI/CD";
- Neste processo, equipes de Desenvolvimento e Operações trabalham juntas, fazendo uso de metodologias ágeis, com abordagem tanto para DevOps ou SRE's (Site Reliability Engineering).

Importância do CI/CD no Desenvolvimento das Aplicações

CI/CD ajuda as equipes de Desenvolvimento, Segurança e Operações a trabalhar da maneira mais eficiente e eficaz possível. Mitigando o trabalho manual e liberando as equipes de DevOps para serem mais inovadoras no desenvolvimento de software.

A automação torna os processos previsíveis e repetíveis para que haja menos oportunidades de erros, decorrentes da intervenção humana.

LINK

LINK

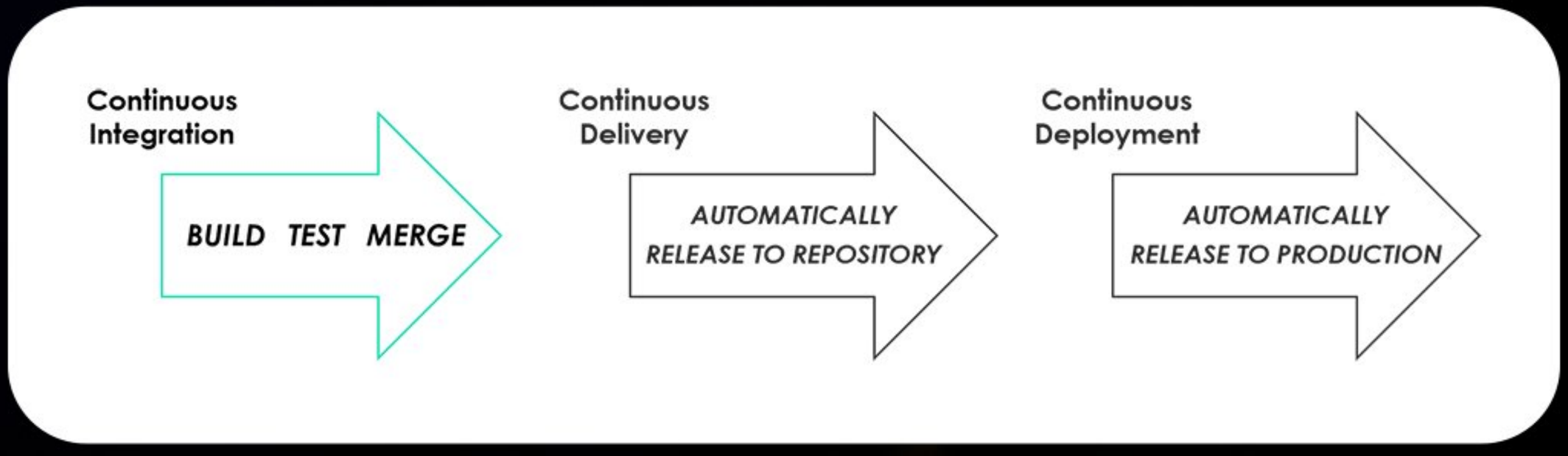
Benefícios da Automação na Entrega das Aplicações

LINK

- Garantia de ‘Compliance’;
- Reduz o tempo de trabalho;
- Entregas com mais frequencias;
- Qualquer membro da equipe será capaz de fazer ‘deploys’;
- As implantações tornam-se muito menos propensas a erros e muito mais repetíveis.

Continuous Integration (CI)

Continuous Integration (CI) é a prática de automatizar a integração das modificações em códigos de vários contribuidores, em um único projeto de software. É uma das boas práticas recomendadas do DevOps, permitindo que os desenvolvedores mesclem frequentemente alterações de código em um repositório central.

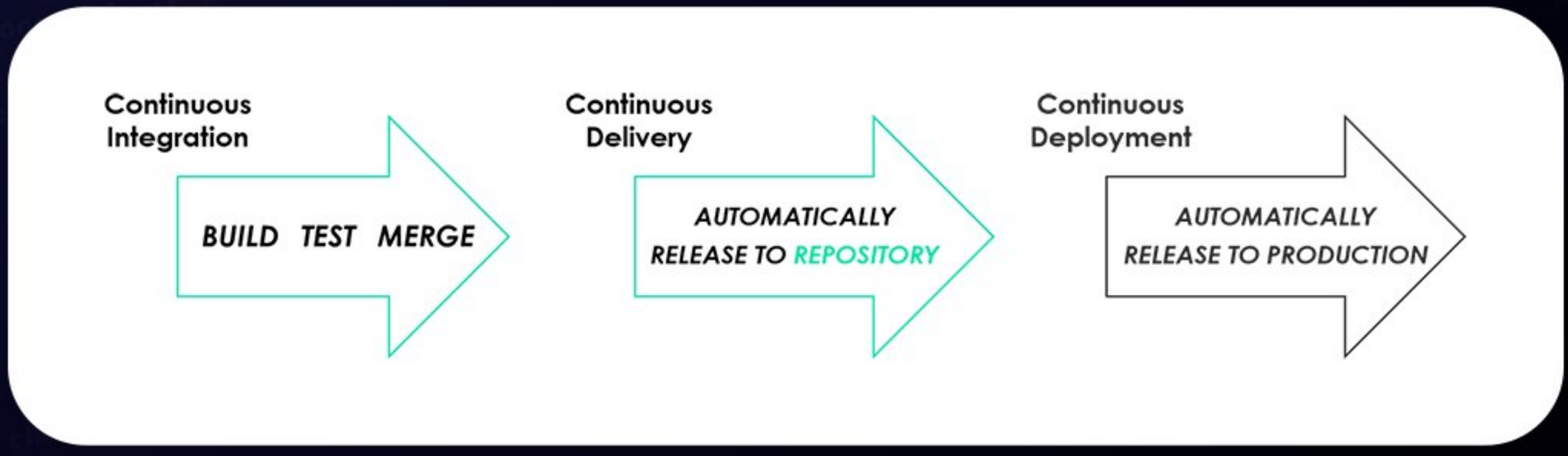


LINK

Continuous Delivery (CD)

Continuous Delivery(CD) é uma extensão do 'Continuous Integration (CI)' para implementar todas as alterações feitas no código, como por exemplo: novos recursos, alterações de configuração, correções de bugs, em ambientes de teste/produção.

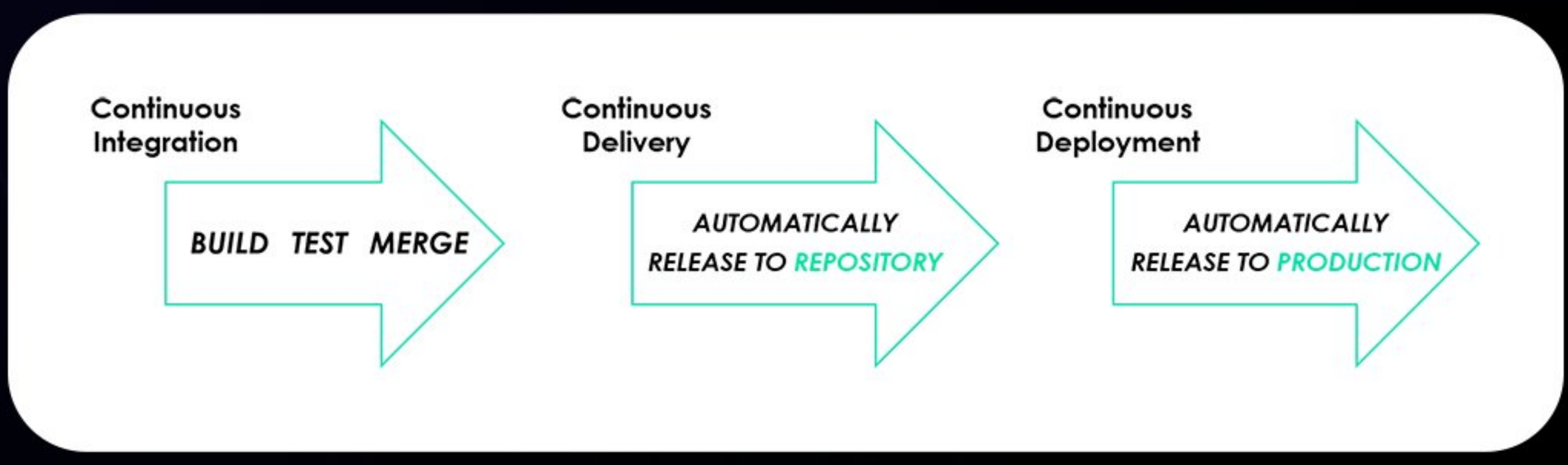
Continuous Delivery (CD) necessita de intervenção humana para implementar o código.



Continuous Deployment (CD)

Continuous Deployment(CD) mesmo conceito, porém...

Continuous Deployment (CD) não necessita de intervenção humana para implementar o código.



Estágios Pipeline CI/CD

DEVOPS ENGINEER

Source → **Build** → **Test** → **Deploy**

`git push`

- Compile code
- Unit test
- Integration test
- Staging
- QA
- Production

Continuous Integration

Continuous Delivery (manual)

Continuous Deployment (automatic)

Principais Ferramentas CI/CD

Jenkins

- Jenkins é uma ferramenta 'Open-source' de automação;
- Desenvolvido em Java;
- Possui muitos plugins.

circleci

- Fornece suporte e serviços de classe empresarial, com a flexibilidade de uma startup.
- Você pode integrar com GitHub, GitHub Enterprise e Bitbucket;
- Executa as 'builds' usando um contêiner ou máquina virtual.

Travis CI

- Travis CI detecta automaticamente novos commits e envia para um repositório GitHub;
- Suporta muitas configurações de "builds" e linguagens: Node, PHP, Python, Java...;
- Permite 'Deployment' em vários serviços em 'nuvem'.

GitLab

- GitLab é um conjunto de ferramentas para gerenciar diferentes aspectos do ciclo de vida de desenvolvimento de software;
- Você pode criar jobs em uma máquina virtual, em um contêiner Docker, por exemplo.

Serviços 'Cloud-native' CI/CD

AWS CodePipeline

AWS CodePipeline é um serviço totalmente gerenciado de entrega contínua que ajuda a automatizar pipelines oferecendo atualizações rápidas e confiáveis tanto de aplicações, como de infraestruturas.

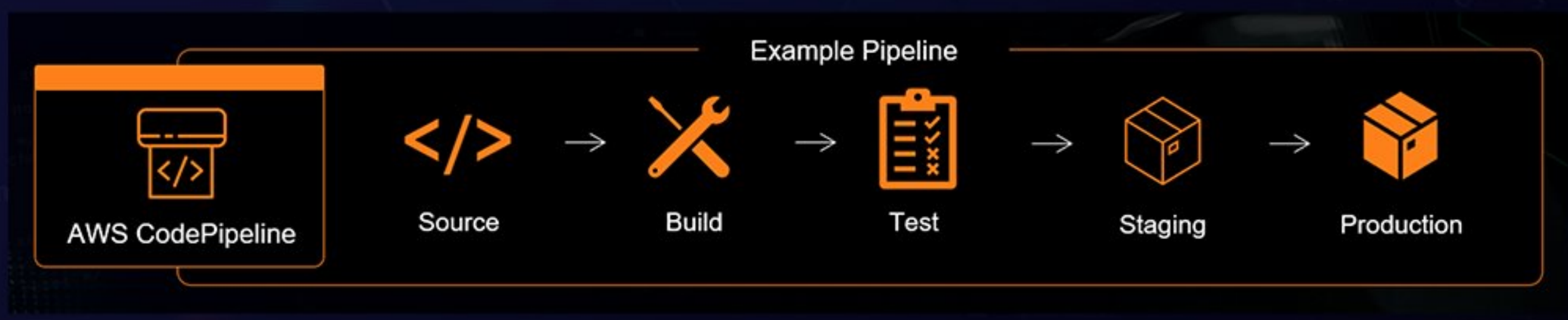
Azure DevOps

Azure DevOps é uma plataforma de software como serviço (SaaS) que fornece práticas e ferramentas de DevOps para o ciclo de vida de software de ponta a ponta. E, o Azure DevOps se integra com a maioria das outras ferramentas DevOps disponíveis no mercado.

Google Cloud Build

O Cloud Build é um serviço que executa suas "builds" na Google Cloud e permite importar código-fonte de uma variedade de repositórios, executando conforme suas especificações e produzindo artefatos como contêineres Docker ou arquivos Java, por exemplo.

CI/CD na AWS



AWS CodeCommit é um serviço de controle de versionamento de código-fonte totalmente gerenciado, seguro e altamente escalável que hospeda repositórios privados do Git.

AWS CodeBuild é um serviço de integração contínua (CI) totalmente gerenciado que compila código-fonte, executa testes e produz pacotes (artefatos) de software prontos para implantação (deploy - CD).

AWS CodeDeploy é um serviço de "deployment" totalmente gerenciado que automatiza implantações de software em vários serviços de computação, como instâncias EC2, Elastic Container Service (ECS), AWS Lambda, e também em servidores on-premises.

Breve história do Jenkins

2004 O projeto Jenkins foi iniciado, originalmente chamado de 'Hudson' por Kohsuke Kawaguchi, enquanto ele trabalhava para a 'Sun Microsystems'.

2011 A Oracle, proprietária da Sun Microsystems, teve uma disputa com a comunidade de código aberto Hudson, então eles deram 'fork' no 'Hudson' e o renomearam para 'Jenkins'.

Today Jenkins é a solução mais amplamente adotada para entrega contínua, graças à sua extensibilidade e a uma comunidade vibrante e ativa. A comunidade Jenkins oferece mais de 1.700 plugins.

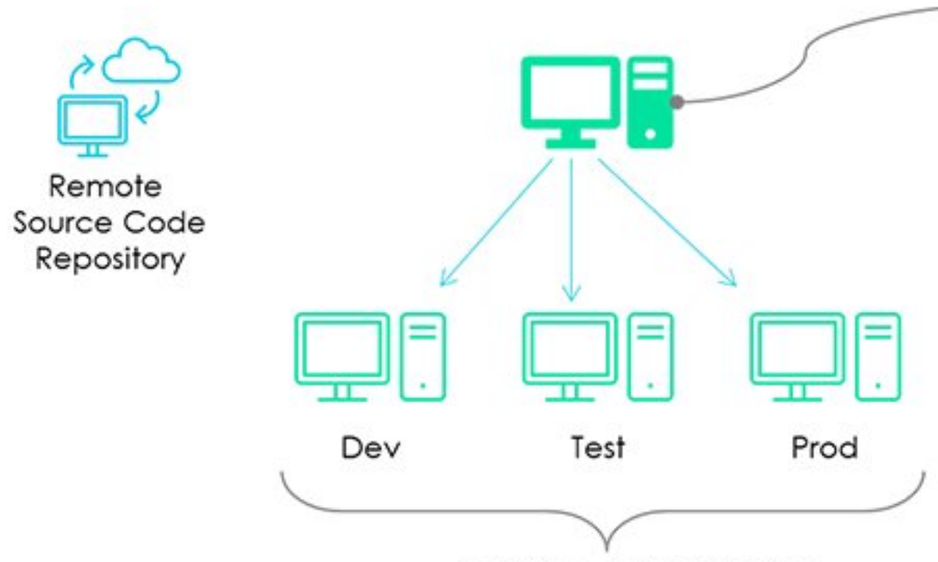
Jenkins

Um mordomo (butler) pode realizar suas tarefas de acordo com seu pedido.

Princpais Recursos do Jenkins

- **Free Open Source** — Jenkins é uma ferramenta de código aberto apoiado por uma forte e grande comunidade.
- **Easy Installation** — Jenkins é um programa independente baseado em Java, pronto para ser executado em ambientes Windows, Mac OS e sistemas operacionais Unix (Linux).
- **Easy Configuration** — Jenkins é facilmente instalado e configurado usando sua interface web, com excelentes funcionalidades e dashboard para acompanhamento dos Jobs.
- **Easy Distribution** — Jenkins pode distribuir 'Jobs', 'tasks' facilmente em várias máquinas para compilações, testes e implantações em várias plataformas.
- **Available Plugins** — Existem centenas de plug-ins disponíveis no "Update Center", integrando-se a todas as ferramentas do universo CI/CD.

Jenkins Arquitetura: Master & Agent Nodes



Jenkins Master é o servidor Jenkins primário.

Executa uma infinidade de tarefas, como:

- Agendamento de "job"/"builds";
- Registra e apresenta os resultados das "builds";
- Entrega das "builds" para os 'nós' agentes;
- Monitoramento dos 'agentes'.

Jenkins Master envia diversas tarefas para os "Agent Nodes" executarem.

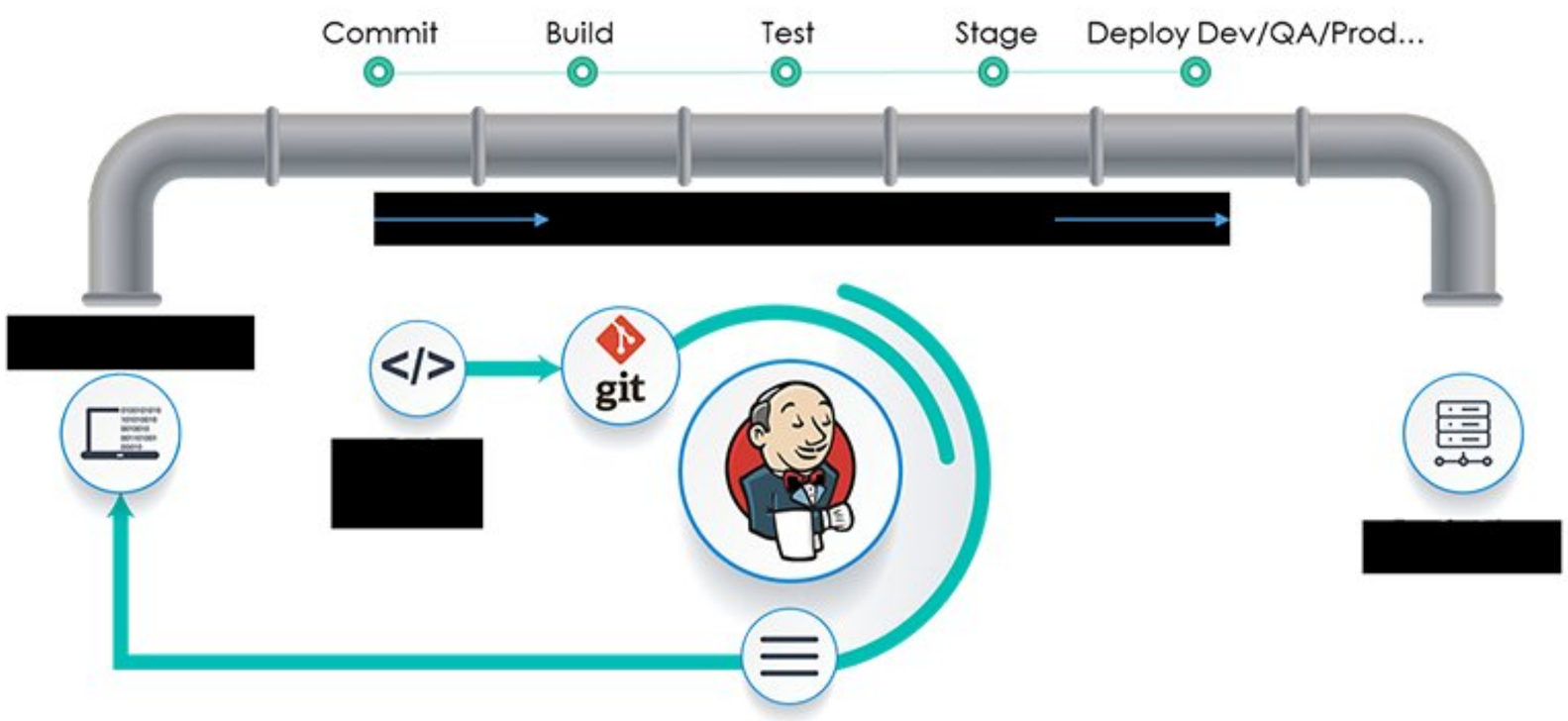
O que é 'Jenkins pipeline'?

Jenkins Pipeline é um conjunto de plug-ins para implementação e integração de pipelines de entrega contínua no Jenkins.

Fornece um conjunto de ferramentas para modelar pipelines seja de entrega simples ou complexa.

A definição normalmente é escrita em um arquivo de texto (chamado Jenkinsfile)

que é validado no repositório de controle de origem do projeto.



Sintaxe: Declarative vs. Scripted Pipeline

```
pipeline{
    agent any
    stages{
        stage('Build'){
            steps{
                //
            }
        }
        stage('Test'){
            steps{
                //
            }
        }
        stage('Deploy'){
            steps{
                //
            }
        }
    }
}
```

```
node {
    stage('Build') {
        //
    }
    stage('Test') {
        //
    }
    stage('Deploy') {
        //
    }
}
```

Pipeline

Define todo o seu processo de ‘Build’, que normalmente inclui etapas (estágios) como ‘building’, ‘testing’ e ‘delivering’.

Agent

Execute a ‘Pipeline’ ou qualquer uma de suas etapas (estágios), em qualquer agente disponível.

Stage

Define um subconjunto distinto de tarefas a serem executadas referente a uma determinada Pipelines. É usado por muitos plug-ins para visualizar ou apresentar o status/progresso do ‘Jenkins Pipeline’.

Step

“Simples Tarefa”. “Step” informa ao Jenkins o que fazer em uma determinada etapa do processo. Por exemplo, para executar o comando ‘make’, usando o “sh step”: sh 'make'.

Node

É um host que faz parte do ambiente Jenkins, capaz de executar uma Pipeline.

LINK

Declarative Pipeline

- Inicia com o bloco 'pipeline’;
- Utiliza uma sintaxe estruturada;
- Possui validação de código de pipeline;
- Permite reiniciar a partir de um estágio específico;
- Permite pular alguma etapa usando o bloco ‘when’;
- É uma funcionalidade mais recente e avançada;
- Possui recursos como blocos de “option” e “environment”;
- Projetado para facilitar escrita e leitura do código da Pipeline.

Scripted Pipeline

- Inicia com o bloco 'node';
- Foi a primeira e tradicional implementação do pipeline como código em Jenkins;
- Projetada como uma DSL (Domain Specific Language) de uso geral, e construída com Groovy;
- Scripted Pipeline segue um modelo de programação mais imperativo, e você deve especificar o que deseja e como deve ser feito.

Pipeline pode ser escrita usando ambas sintaxes e você pode executar as mesmas etapas, mas 'declarative' possui mais opções do que a 'scripted'.

LINK

Entendendo o Arquivo 'Jenkinsfile'

Jenkins Pipeline pode ser escrito em um arquivo de texto chamado 'Jenkinsfile' (você pode usar um nome diferente), que é enviado ao repositório do projeto.


```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                echo 'Building..'
            }
        }
        stage('Test') {
            steps {
                echo 'Testing..'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying....'
            }
        }
    }
}
```

Benefícios do Jenkinsfile

- Histórico | Versionamento;
- Auditoria e Rastreabilidade da Pipeline;
- Revisão/iteração de código no Pipeline;
- Suporta ambas sintaxes: Declarative e Scripted;
- Pipeline com uma única fonte/origem, que pode ser visualizada e editada por vários membros do projeto.

LINK

#PraCima