DevOps

Introdução ao DevOps

O que é DevOps?

Desenvolvimento (**dev**elopment) + Operações (**op**eration**s**)

DEV - time responsável pelo desenvolvimento da aplicação

OPS - time responsável por garantir estabilidade, disponibilidade, segurança e desempenho.

- Infraestrutura
- DBA
- Segurança da informação
- Suporte/Help desk
- Monitoramento e Observabilidade

O que é DevOps?

DevOps é um conjunto de práticas, ferramentas e uma filosofia cultural que automatizam e integram os processos entre o desenvolvimento de software e as equipes de TI. Ele enfatiza o empoderamento da equipe, a comunicação e colaboração entre equipes e a automação da tecnologia.

Práticas adotadas

- Integração contínua
  - Refere-se à prática de integrar código de forma frequente em um repositório compartilhado, onde testes automatizados são executados para garantir que novas alterações não quebrem o sistema
- Entrega contínua
  - É a extensão da integração contínua, onde o código é automaticamente preparado para ser lançado em produção, permitindo que as equipes entreguem novas versões de software de forma rápida e confiável
- Micro Serviços
  - Uma abordagem arquitetônica que divide aplicações em pequenos serviços independentes, facilitando a implementação e a escalabilidade
- Infraestrutura como codigo
  - Permite que a infraestrutura de TI seja gerenciada e provisionada através de código, facilitando a automação e a consistência
- Monitoramento e registro de logs
  - Ferramentas que permitem o acompanhamento do desempenho das aplicações e a coleta de dados para análise, ajudando a identificar problemas rapidamente

Benefícios

- Velocidade
  - Com a automação e a integração contínua, as equipes podem lançar atualizações e novos recursos com mais frequência, respondendo rapidamente às demandas do mercado.
- Confiabilidade
  - A automação de testes e processos reduz a probabilidade de erros, garantindo que as entregas sejam mais confiáveis.
- Escalabilidade
  - A infraestrutura automatizada permite que as empresas escalem suas operações de forma eficiente, adaptando-se rapidamente às mudanças na demanda.
- Colaboração melhorada
  - O DevOps promove uma cultura de colaboração entre as equipes de desenvolvimento e operações, resultando em um fluxo de trabalho mais harmonioso.
- Segurança
  - A integração de práticas de segurança no ciclo de desenvolvimento (DevSecOps) garante que a segurança seja uma prioridade desde o início, reduzindo riscos e vulnerabilidades

DevOps x Modelos tradicionais

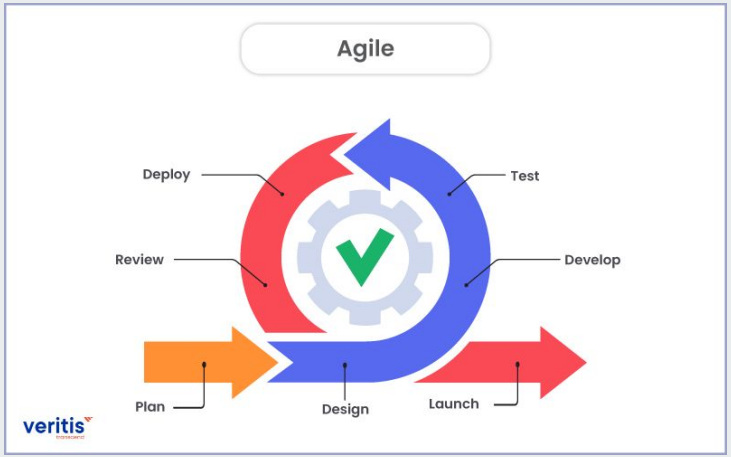
Waterfall (cascata)

Um modelo de produção linear permite que os desenvolvedores prossigam após a conclusão da fase anterior

DevOps x Modelos tradicionais

Agile

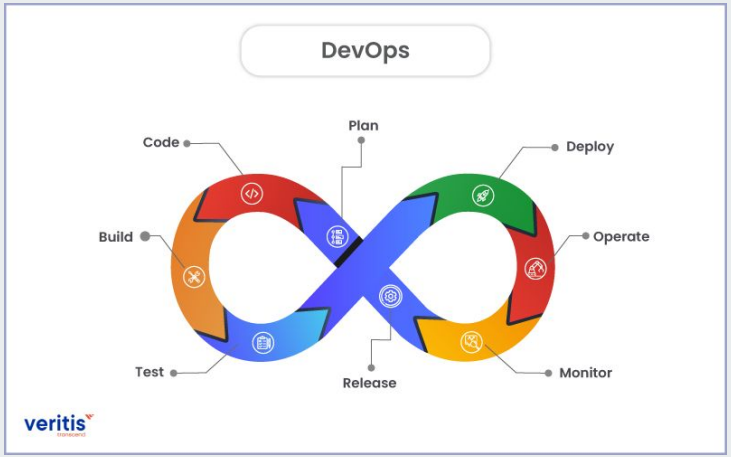
Um modelo de produção onde interações podem ser implementadas com frequência, entregando pequenas melhorias e atualizações



DevOps x Modelos tradicionais

DevOps

Um modelo de produção que integra as melhores ideias do Agile com a equipe de operações



Prática - Git

O que é Git e para que serve?

O que é?

- Git é um sistema de controle de versão distribuído.
- Criado por Linus Torvalds em 2005 para gerenciar o código fonte do Linux.
- Permite acompanhar alterações no código, colaborando de forma eficiente.
- Usado por desenvolvedores para rastrear histórico e reverter mudanças quando necessário.

Para que serve?

- Gerenciar versões do código sem risco de perder informações.
- Permitir colaboração entre múltiplos desenvolvedores.
- Criar branches para desenvolver novas funcionalidades sem impactar o código principal.
- Facilitar rollback para versões anteriores se algo der errado.

Diferença entre Git e Github

Git

- Ferramenta de controle de versão instalada localmente.
- Permite rastrear mudanças no código de forma offline.
- Utiliza repositórios locais e remotos para sincronizar arquivos.

Github

- Plataforma online para armazenar e compartilhar repositórios Git.
- Facilita o trabalho colaborativo com pull requests e code reviews.
- Inclui funcionalidades extras como Actions (CI/CD), Issues (gestão de tarefas) e Wikis.

Git é a ferramenta de versionamento, Github plataforma onde os repositórios Git podem ser armazenados e compartilhados.

Por que versionamento de código é essencial no DevOps?

- Facilita a colaboração entre desenvolvedores e times de operação.
- Mantém histórico detalhado de todas as alterações.
- Permite reversão rápida em caso de bugs ou falhas.
- Integração com pipelines de CI/CD para automação de testes e deploy.
- Garante rastreabilidade e controle de qualidade no desenvolvimento de software.

Comandos

```
$ git --version

$ git config --global user.name “First name Last name”

$ git config --global user.email “email@satc.edu.br”

$ git config --list
```

O que é um repositório?

- Um repositório é um diretório que armazena arquivos e seu histórico de versões.
- Contém código-fonte, documentação e arquivos de configuração do projeto.
- No Git, um repositório pode ser local ou remoto.

Comandos

- Criar repositorio http://github.com
- Clicar em “New repository”
  - Repository name: unisatc-devops
  - Description: “Repositório de testes para a disciplina de DevOps”
  - Marcar “Public” e “Add a README file”
  - Clicar em “Create repository”

Clonando o repositório

- Clonar o repositório localmente
  - No Github, clicar no botão “Code” e copiar o link “Https”
  - No terminal rodar:
    - `$ git clone https://github.com/seu_usuario/unisatc-devops.git`
    - `cd unisatc-devops`
- Criar um novo arquivo
  - Criar um arquivo chamado index.txt dentro do repositório local:
    - `$ echo “Meu primeiro commit no Github!” > index.txt`

Adicionando, comitando e enviando alterações

- Adicionar o arquivo no controle de versão
  - `$ git add index.txt`
- Criar um commit com uma mensagem descritiva
  - `$ git commit -m “adicionando meu primeiro arquivo”`
- Enviar alterações para o Github
  - `$ git push origin main`

Adicionando, comitando e enviando alterações

- Adicionar o arquivo no controle de versão
  - `$ git add index.txt`
- Criar um commit com uma mensagem descritiva
  - `$ git commit -m “adicionando meu primeiro arquivo”`
- Enviar alterações para o Github
  - `$ git push origin main`

Avançando - alias

Criando alias

- `$ git config --global --edit`
- `$ git config --global core.editor code`
  - Após entrar no vscode, adicionar a flag --wait no editor
- Adicionar a sessão [alias] com
  - `s = !git status -s`
  - `c = !git add --all && git commit -m`
  - `l = !git log --oneline`
  - `lf = !git log --pretty=format:'%C(blue)%h%C(red)%d %C(white)%s - %C(cyan)%cn, %C(green)%cr'`

Avançando - stash

Para que serve o git stash?

O git stash é um comando que permite salvar temporariamente mudanças não commitadas no seu repositório sem precisar criar um commit.
Útil para quando você precisa alternar rapidamente entre branches ou atualizar seu código sem perder suas alterações atuais.

Avançando - stash

- Guardando mudanças
  - `$ git stash`
- Visualizando stashs salvos
  - `$ git stash list`
- Recuperando stash específico
  - `$ git stash apply stash@{0}`
- Removendo stash manualmente
  - `$ git stash drop stash@{0}`
- Limpando todos os stash
  - `$ git stash clear`