Aula 1

**\*\*Título da Aula:\*\*** Introdução à Cultura DevOps

**\*\*Objetivos de Aprendizagem:\*\***

1. **\*\*Compreender o Conceito de DevOps:\*\***

- Justificativa: Entender que DevOps é mais do que ferramentas; é uma cultura que integra desenvolvimento e operações para acelerar a entrega de software. No mercado de trabalho, isso é crucial para as empresas que buscam eficiência e inovação contínua.

1. **\*\*Aprender Sobre Automação e Integração Contínua:\*\***

- Justificativa: Os alunos devem aprender a automatizar processos para reduzir o tempo de entrega e aumentar a confiabilidade. Isso é uma habilidade valorizada em ambientes empresariais que buscam melhorar a eficiência operacional.

1. **\*\*Entender os Princípios de Colaboração e Melhoria Contínua:\*\***

- Justificativa: Colaboração eficaz entre equipes de desenvolvimento e operações é essencial para o sucesso do DevOps, e a melhoria contínua garante que os processos se tornem cada vez mais eficientes e adaptáveis a mudanças.

1. **\*\*Familiarizar-se com Tecnologias Modernas:\*\***

- Justificativa: Conhecer tecnologias como Docker, Kubernetes, e infraestrutura como código é importante para entender a arquitetura de sistemas modernos. No mercado, isso prepara os alunos para trabalhar em projetos de larga escala e com tecnologias de ponta.

1. **\*\*Desenvolver Habilidades Práticas em DevOps:\*\***

- Justificativa: A prática de configurar pipelines de integração e entrega contínua prepara os alunos para desafios reais no mercado, onde a capacidade de implementar soluções eficazes e automatizadas é altamente demandada.

**\*\*Principais Tópicos Cobertos:\*\***

**- \*\*Conceitos Gerais:\*\***

- Introdução a DevOps como cultura e metodologia.

- Benefícios do DevOps e sua importância no mercado.

**- \*\*Componentes:\*\***

- Princípios de automação, colaboração, medição e melhoria contínua.

- Ferramentas e tecnologias relevantes como Docker, Git, Jenkins, e infraestrutura como código.

**- \*\*Evolução e Contexto:\*\***

- Evolução das metodologias de desenvolvimento: Waterfall, Agile, e a introdução do DevOps.

- Comparação entre monolitos e micro serviços.

**- \*\*Exemplos Práticos:\*\***

- Dinâmica de criação de um "artefato analógico de software" para exemplificar a importância da automatização e colaboração.

- Exposição dos benefícios de uma pipeline automatizada e da função do DevOps em um cenário prático.

**\*\*Perguntas Guiadas para Estudo (Elaboração Intercalada):\*\***

1. O que diferencia DevOps de outras metodologias de desenvolvimento como Agile?
   1. DevOps integra desenvolvimento e operações para acelerar a entrega de software, enquanto Agile foca apenas na adaptação rápida durante o desenvolvimento.
2. Como a automação pode impactar a eficiência de uma equipe de desenvolvimento?
   1. A automação reduz tarefas repetitivas, diminuindo erros e permitindo que a equipe se concentre em atividades mais valiosas.
3. Quais são os benefícios de utilizar micro serviços em vez de um monolito?
   1. Permitem atualização e escalabilidade independentes, tornando o sistema mais flexível e responsivo a mudanças.
4. Como a infraestrutura como código pode facilitar o trabalho de um DevOps?
   1. Automatiza a configuração de ambientes, garantindo consistência e facilidade de replicação e escalabilidade.
5. Quais são os desafios enfrentados por equipes que não implementam DevOps?
   1. Equipes sem DevOps enfrentam comunicação falha, lançamentos lentos e maior risco de erros em produção.
6. Como a colaboração entre desenvolvimento e operações pode ser melhorada em uma organização?
   1. Ferramentas de CI/CD e comunicação aberta ajudam a integrar desenvolvimento e operações, eliminando barreiras.
7. De que maneira a medição e o monitoramento contribuem para a melhoria contínua?
   1. Fornecem dados para identificar melhorias, corrigir falhas e otimizar processos constantemente.

Aula 2

- Linux

- Docker / Docker Compose

- Git

**\*\*Título da Aula:\*\***Ferramentas DevOps e Configuração de Ambiente

**\*\*Objetivos de Aprendizagem:\*\***

1. **\*\*Configuração de Ambiente Linux e Ferramentas DevOps:\*\***

Os alunos devem ser capazes de instalar e configurar o WSL (Windows Subsystem for Linux), Docker, Docker Compose e Git em seus computadores. Isso é essencial, pois o mercado de trabalho valoriza profissionais que conseguem configurar seus próprios ambientes de desenvolvimento de forma independente.

1. **\*\*Compreensão de Tecnologias DevOps:\*\***

Identificar e entender as tecnologias e dependências utilizadas em uma aplicação DevOps, como o Node.js e bancos de dados relacionados. No mercado de trabalho, é vital que os profissionais compreendam as ferramentas que estão utilizando, pois isso afeta diretamente a eficácia e eficiência dos processos DevOps.

1. **\*\*Manipulação de Repositórios Git:\*\***

Os alunos devem aprender a clonar, versionar e subir códigos para repositórios Git usando chaves SSH. O Git é uma ferramenta fundamental no desenvolvimento de software moderno, e a habilidade de gerenciar repositórios é crucial para qualquer profissional da área.

1. **\*\*Execução de Comandos Básicos no Linux:\*\***

Os alunos devem ser capazes de utilizar comandos básicos do Linux para navegar e manipular arquivos e pastas, habilidades necessárias para qualquer desenvolvedor ou engenheiro DevOps.

**\*\*Principais Tópicos Cobertos:\*\***

**- \*\*Conceitos Gerais:\*\***

- Importância da configuração de ambiente no desenvolvimento DevOps.

- Introdução à aplicação NodeGoot como um exemplo prático.

**- \*\*Componentes:\*\***

- Instalação e configuração do WSL.

**- Instalação do Docker, Docker Compose e Git.**

**- \*\*Configuração de Chaves SSH:\*\***

- Geração e configuração de chaves SSH para acesso seguro a repositórios Git.

**- \*\*Manipulação de Repositórios:\*\***

- Comandos básicos de Git: clone, add, commit, push e configuração de chaves SSH.

**- \*\*Comandos Linux:\*\***

- Utilização de comandos básicos no terminal Linux:

Touch readme.txt -> cria um arquivo txt

Nano readme.txt -> abre o arquivo txt

ls, cd, cp, cat, mkdir, rm.

**\*\*Exemplos Práticos:\*\***

**- \*\*Configuração do Ambiente:\*\***

Os alunos foram guiados na instalação do WSL em seus computadores, seguido pela instalação do Docker, Docker Compose e Git. Isso incluiu o uso de comandos de terminal para configurar o ambiente Linux.

**- \*\*Manipulação do Git:\*\***

Foi demonstrado como configurar chaves SSH, clonar repositórios, adicionar arquivos ao versionamento e subir alterações usando comandos Git.

**- \*\*Navegação e Manipulação de Arquivos no Linux:\*\***

Os alunos praticaram comandos de terminal para criar e manipular arquivos e pastas no Linux, como cp para copiar arquivos recursivamente e cat para visualizar o conteúdo de arquivos.

**\*\*Perguntas Guiadas para Estudo:\*\***

1. Quais são os passos para instalar o WSL e por que ele é importante no ambiente DevOps?
   1. Abra o PowerShell como Administrador e execute wsl --install. Ele permite rodar Linux no Windows, essencial para DevOps, pois muitas ferramentas DevOps são feitas para Linux.
2. Como configurar uma chave SSH e qual é a sua importância ao trabalhar com repositórios Git?
   1. No terminal, use ssh-keygen, copie a chave pública para o Git. SSH é importante porque permite conexões seguras e automáticas com repositórios, sem pedir credenciais constantemente.
3. Quais são os comandos básicos para manipulação de arquivos e pastas no Linux?
   1. ls (listar), cd (navegar), mkdir (criar pasta), cp (copiar), rm (remover) e cat (ver conteúdo) são fundamentais para gerenciar arquivos e pastas no ambiente DevOps.
4. O que é o Docker e como ele se integra no fluxo de trabalho DevOps?
   1. Cria contêineres com tudo que a aplicação precisa, facilitando o desenvolvimento e o CI/CD, pois permite rodar ambientes consistentes e replicáveis.
5. Como clonar um repositório Git usando SSH e quais são os benefícios dessa abordagem?
   1. Use git clone git@github.com:usuario/repo.git. SSH é mais seguro e evita pedir senhas, ideal para automação.

Aula 3

**\*\*Título da Aula:\*\***Princípios de DevOps: Colaboração e Automação

**\*\*Objetivos de Aprendizagem:\*\***

1. **\*\*Compreender os comandos básicos do Linux:\*\***

Fundamental para automações e navegação em ambientes de desenvolvimento, habilidades muito requisitadas no mercado de trabalho para desenvolver scripts e gerenciar servidores.

1. **\*\*Implementar scripts de automação com Shell Script:\*\***

Capacita o aluno a automatizar tarefas repetitivas e orquestrar deploys de aplicações, uma habilidade desejável em DevOps para aumentar a eficiência e reduzir erros humanos.

1. **\*\*Desenvolver e automatizar o deploy de uma aplicação Flask com front-end em HTML/CSS/JS:\*\***

Promove a compreensão prática de como os componentes DevOps interagem desde o desenvolvimento até a entrega contínua, refletindo demandas reais do mercado.

1. **\*\*Colaborar em equipes para atingir objetivos comuns:\*\***

Reflete a necessidade de trabalhar em times diversos, uma prática constante em ambientes de trabalho modernos e ágeis.

1. **\*\*Resolver problemas de comunicação entre front-end e back-end usando CORS:\*\***Ensina a lidar com problemas de integração, essencial para garantir que aplicações web funcionem corretamente em ambientes reais.

**\*\*Principais Tópicos Cobertos:\*\***

**- \*\*Conceitos Gerais:\*\***

- Importância da colaboração em DevOps.

- Objetivo de entregar valor ao cliente final.

**- \*\*Componentes:\*\***

- Ferramentas DevOps: Git, Docker, Docker Compose.

- Comandos básicos do Linux (CD, LS, CP, MV).

- Linguagens para automação: Shell Script, Python, JavaScript.

**- \*\*Automação:\*\***

- Criação de scripts em Shell Script para automação de tarefas.

- Uso de comandos Linux para criar, copiar e listar arquivos.

**- \*\*Implementação Prática:\*\***

- Desenvolvimento de uma API com Flask.

- Criação de uma interface web utilizando HTML, CSS e JavaScript.

- Configuração de um ambiente virtual de Python.

**- \*\*Colaboração Prática:\*\***

- Trabalho em equipes para desenvolver e integrar componentes de software.

**\*\*Exemplos Práticos:\*\***

- Criação de um script em Shell Script para automatizar a criação de um arquivo de configuração e estrutura de diretórios.

- Desenvolvimento de uma API Flask que soma dois números e retorna o resultado em JSON.

- Implementação de uma interface web para consumir a API Flask.

**\*\*Perguntas Guiadas para Estudo (Elaboração Intercalada):\*\***

1. Quais são os comandos básicos do Linux necessários para navegação de diretórios?
   1. cd (mudar diretório), ls (listar conteúdo), pwd (mostrar caminho atual), mkdir (criar diretório), rmdir/rm -r (remover diretório).
2. Explique como o Shell Script pode ser usado para automatizar tarefas no desenvolvimento de software.
   1. **Shell Script para automação:** Facilita tarefas repetitivas como deploys e configurações, aumentando a eficiência e reduzindo erros.
3. Como a colaboração entre equipes de desenvolvimento e operações contribui para o sucesso da metodologia DevOps?
   1. Melhora a comunicação e eficiência, permitindo entregas rápidas e confiáveis com menor chance de problemas.
4. Quais são os passos necessários para criar um ambiente virtual Python e por que ele é importante?
   1. python3 -m venv nome e ativar. Isola dependências, evitando conflitos entre projetos.
5. Descreva o problema de CORS e como ele pode ser resolvido em um projeto web.
   1. Navegadores bloqueiam requisições entre domínios; resolvido configurando o servidor para permitir origens específicas.
6. Como o uso de Git se integra ao processo de automação de deploy?
   1. Git integra com CI/CD para deploys automáticos, agilizando atualizações seguras.