INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE

INFORMÁTICA PARA INTERNET

MICKAELLE KARINE SOUZA SILVA

INFRAESTRUTURA COMO CÓDIGO

Atividade de Pesquisa

Caicó/RN

2025

1. Infraestrutura como Código (IaC)

A Infraestrutura como Código (IaC) é a prática de gerenciar e provisionar a infraestrutura de TI através de arquivos de configuração e código, em vez de processos manuais. Isso permite tratar a infraestrutura da mesma forma que o código de uma aplicação, incluindo versionamento, testes e implantações automáticas.

1.1 Principais Características:

- **Automação**: Reduz erros humanos e acelera o provisionamento, eliminando configurações manuais.
- **Versionamento**: O código é armazenado em sistemas de controle de versão (como Git), permitindo rastreamento e colaboração.
- **Consistência e Padronização**: Garante que os ambientes (desenvolvimento, teste, produção) sejam idênticos, evitando desvios de configuração.
- **Idempotência**: A aplicação repetida do mesmo código resulta no mesmo estado, independentemente do ambiente inicial.

1.2 Abordagens:

- **Declarativa**: Descreve o "estado desejado" da infraestrutura.
- **Imperativa**: Especifica os "passos exatos" para configuração.

1.3 Reusabilidade:

Módulos e templates podem ser reutilizados para provisionar infraestrutura similar em diferentes projetos ou ambientes.

2.Impacto da IaC no DevOps

A Infraestrutura como Código (IaC) é essencial para a cultura e práticas de DevOps, promovendo colaboração, agilidade e confiabilidade entre equipes de desenvolvimento e operações.

2.1 Principais Impactos:

- **Aceleração das Entregas**: Permite provisionar ambientes rapidamente, reduzindo o tempo de "time-to-market" de dias ou semanas para minutos.
- **Consistência e Redução de Erros**: Automatiza o provisionamento, eliminando inconsistências e erros manuais, resultando em implantações mais confiáveis.
- **Colaboração Aprimorada**: O código da infraestrutura pode ser versionado e revisado, facilitando a colaboração entre desenvolvedores e operações.
- **Integração Contínua/Entrega Contínua (CI/CD)**: A IaC se integra aos pipelines de CI/CD, permitindo testes e implantações automáticas, alinhando a infraestrutura com as necessidades da aplicação.
- **Escalabilidade e Elasticidade**: Facilita o dimensionamento da infraestrutura conforme a demanda, permitindo respostas rápidas e otimização de custos.
- **Infraestrutura Imutável**: Promove ambientes imutáveis, onde mudanças requerem a criação de novos ambientes, garantindo maior estabilidade.
- **Recuperação de Desastres e Testes**: Facilita a recriação de ambientes em caso de desastre e a execução de testes de forma automatizada e repetível.

3. Diferenças entre Docker, Podman e Kubernetes

Docker, Podman e Kubernetes são ferramentas fundamentais no ecossistema de contêineres, cada uma com funções distintas.

3.1 Docker

- **O que é**: Plataforma de contêineres que empacota aplicações e suas dependências em contêineres isolados.
- **Arquitetura**: Funciona com uma arquitetura cliente-servidor, onde o Docker Daemon gerencia contêineres e a interação é feita via Docker CLI.
- **Características**:
    - **Imagens Docker**: Templates somente leitura para criar contêineres.
    - **Contêineres**: Instâncias executáveis de imagens.
    - **Docker Hub**: Registro público para compartilhar imagens.
    - **Docker Compose**: Para aplicações multi-contêineres.

- **Docker Swarm**: Ferramenta de orquestração nativa.
- **Uso principal**: Desenvolvimento local e execução de aplicações em contêineres.

3.2 Podman

- **O que é**: Ferramenta de linha de comando para gerenciar contêineres e pods, compatível com a CLI do Docker.
- **Arquitetura**: Daemonless, interage diretamente com o runC, aumentando a segurança e reduzindo o consumo de recursos.
- **Características**:
  - Daemonless: Não requer um daemon em execução.
  - Rootless: Executa contêineres sem privilégios de root.
  - Compatibilidade com Docker CLI: Comandos semelhantes facilitam a transição.
  - Suporte a Pods: Gerencia grupos de contêineres.
  - Geração de manifestos Kubernetes: Cria arquivos YAML compatíveis com Kubernetes.
- **Uso principal**: Ambientes de desenvolvimento e teste em sistemas Linux, priorizando segurança e leveza.

3.3 Kubernetes

- **O que é:** Sistema de orquestração de contêineres projetado para automatizar a implantação e gerenciamento de aplicações em contêineres.
- **Arquitetura:** Opera em um cluster de máquinas com um plano de controle e nós de trabalho.
- **Características**:
  - Orquestração de contêineres: Gerencia implantação, escalabilidade e autorrecuperação.
  - Pods: Menor unidade implantável, contendo um ou mais contêineres.
  - Deployments: Define como os Pods devem ser executados.
  - Services: Facilita a comunicação entre Pods.
  - Namespaces: Organiza recursos logicamente.
  - Auto-scaling: Escala automaticamente com base na carga.

    - Self-healing: Reinicia contêineres com falha e remove os que não respondem.
- **Uso principal**: Gerenciamento de aplicações em contêineres em produção, especialmente para microsserviços e cargas de trabalho complexas.

4. Referências

Atlassian. ([s.d.]). *Kubernetes vs. Docker*. Atlassian. Recuperado 12 de junho de 2025, de https://www.atlassian.com/br/microservices/microservices-architecture/kubernetes-vs-docker

What is the difference between dockers and kubernetes? ([s.d.]). Palo Alto Networks. Recuperado 12 de junho de 2025, de https://www.paloaltonetworks.com/cyberpedia/kubernetes-docker

Entendendo a IaC (Infraestrutura como Código). ([s.d.]). Com.br. Recuperado 12 de junho de 2025, de https://www.rocketseat.com.br/blog/artigos/post/entendendo-a-iac

Aleksic, M. (2023, dezembro 12). *Podman vs. Docker: Everything you need to know*. Knowledge Base by phoenixNAP; phoenixNAP. https://phoenixnap.com/kb/podman-vs-docker