ORIENTAÇÕES

Entrega de Trabalho

1. TÍTULO:

Entrega contínua de uma plataforma de pedidos em microsserviços: do Docker Compose ao Kubernetes com observabilidade e CI/CD

2. DESAFIO:

Você foi contratado(a) como Cloud DevOps Engineer por uma empresa de varejo digital (e-commerce) de médio porte chamada Loja Veloz. Nos últimos 12 meses, a Loja Veloz cresceu rápido e passou a sofrer com problemas recorrentes no ambiente de produção: indisponibilidades durante deploys, dificuldades para escalar em picos de acesso e baixa rastreabilidade de falhas entre serviços.

Contexto (lide: o quê, quem, quando, onde, por quê)

- O quê: Modernização e operação de uma aplicação distribuída baseada em contêineres e microsserviços, com pipeline de entrega contínua e observabilidade.
- Quem: Equipe de produto (devs), operações (SRE/ops) e você, responsável por propor a estratégia Cloud DevOps.
- Quando: A empresa precisa apresentar um plano e um MVP funcional em 4 semanas, pois haverá uma campanha promocional com previsão de pico de tráfego.
- Onde: Ambiente cloud-native (Kubernetes gerenciado ou cluster próprio), com desenvolvimento local padronizado.
- Por quê: O modelo atual causa tempo de entrega alto, deploys arriscados e dificuldade de diagnóstico quando algo falha.

Cenário técnico atual

ORIENTAÇÕES

Entrega de Trabalho

A aplicação “Pedidos Veloz” é composta por:

1. API Gateway (HTTP)
2. Serviço de Pedidos (cria/consulta pedidos)
3. Serviço de Pagamentos (integração externa)
4. Serviço de Estoque (reserva e baixa de itens)
5. Banco de dados (PostgreSQL)
6. Mensageria (opcional, se você justificar) para eventos (ex.: “PedidoCriado”)

Hoje, cada time sobe partes do sistema “como dá” em máquinas diferentes. A empresa quer:

- Ambiente local automatizado e reproduzível para desenvolvimento;
- Conteinerização padronizada (imagens enxutas, versionadas, seguras);
- Orquestração em Kubernetes para produção;
- CI/CD com testes e validações antes do deploy;
- Estratégia de deploy que reduza risco (rolling, blue/green ou canary);
- Escalabilidade automática (HPA e/ou VPA, conforme justificativa);
- Configuração e credenciais bem tratadas (ConfigMaps/Secrets) e políticas de segurança;
- Observabilidade avançada: métricas, logs e tracing distribuído (e, se aplicável, service mesh).

Problema central a ser resolvido

Desenhar e implementar uma proposta fim a fim (do dev ao prod) que reduza risco de deploy, melhore tempo de entrega, permita escalar sob demanda e aumente a confiabilidade por meio de automação, governança e telemetria.

ORIENTAÇÕES

Entrega de Trabalho

O seu trabalho deve resultar em um artefato entregue que demonstre:

- Como o time desenvolve localmente (padronização com Compose);
- Como a aplicação é construída, versionada e publicada (imagens e registry);
- Como sobe em Kubernetes com práticas de produção (deploy, service, config, segurança);
- Como a entrega é automatizada (pipeline);
- Como a operação é observável e resiliente (monitoramento e tracing);
- Como a infra pode ser reproduzida (IaC com Terraform, ao menos em nível de esqueleto + justificativa).

3. FONTE DE PESQUISA:

Você deverá pesquisar um exemplo de como outros profissionais ou empresas já resolveram esse desafio. Para isso, utilize fontes primárias e, obrigatoriamente, traga **um exemplo concreto** (um case público, um repositório de referência, uma arquitetura de referência de vendor, um post técnico oficial etc.).

Fontes de pesquisa primária

1. **Documentação oficial do Kubernetes**
    - Tópicos: Pods, Deployments, Services, ConfigMaps, Secrets, HPA, práticas de segurança.
    - Aulas relacionadas: **Unidade 3** (Fundamentos de K8s, HPA/VPA, Config/Secrets, políticas).
2. **Documentação oficial do Docker e Docker Compose**
    - Tópicos: Dockerfile, build, multi-stage, redes/volumes, Compose para ambientes multi-serviço.

Entrega de Trabalho

- Aulas relacionadas: **Unidade 2** (Contêineres e Imagens; Docker Compose; versionamento/registro).

3. **12-Factor App (metodologia cloud-native)**
   - Tópicos: config por ambiente, logs como streams, processos stateless, port binding, disposability.
   - Aulas relacionadas: **Unidade 1** (Princípios cloud-native; ambientes automatizados).
4. **Documentação oficial do Terraform**
   - Tópicos: IaC, módulos, variáveis, state, boas práticas para ambientes.
   - Aulas relacionadas: **Unidade 4** (Infraestrutura como código com Terraform).
5. **Documentação oficial de CI/CD (GitHub Actions ou GitLab CI ou Azure DevOps)**
   - Tópicos: pipelines, jobs, caching, artifacts, gates, secrets, ambientes.
   - Aulas relacionadas: **Unidade 4** (Pipelines automatizados; testes e validação de build).
6. **Documentação oficial de Istio (ou alternativa de service mesh) + OpenTelemetry**
   - Tópicos: telemetria, tracing distribuído, traffic management, mTLS (se aplicável).
   - Aulas relacionadas: **Unidade 3** (Service Mesh; observabilidade e tracing).

ORIENTAÇÕES

Entrega de Trabalho

5. ENTREGÁVEL E DISTRIBUIÇÃO DA PONTUAÇÃO:

Sua entrega final será composta por **3 entregáveis obrigatórios, totalizando 8,0 pontos**. O objetivo é demonstrar **domínio prático em DevOps moderno**, integrando conteinerização, orquestração com Kubernetes, pipelines de CI/CD e estratégias de observabilidade e escalabilidade em ambientes baseados em microserviços.

Parte Teórica (2,0 pontos)

Formato: Relatório técnico em PDF (mín. 2 páginas | máx. 3 páginas)

Apresentar:

- Conceituação de **arquitetura de microserviços** e o papel do DevOps em ambientes cloud-native
- Explicação sobre **conteinerização**, diferenças entre Docker e Kubernetes e quando utilizar cada abordagem
- Fundamentação teórica sobre:
  - Orquestração de containers
  - CI/CD em ambientes distribuídos
  - Conceitos de observabilidade (métricas, logs e traces)
- Justificativa das **principais decisões arquiteturais** adotadas no projeto

Parte Prática (4,0 pontos)

Formato: Repositório do projeto + Relatório técnico (mín. 3 páginas | máx. 6 páginas)

O projeto prático deverá contemplar obrigatoriamente:

🔹 Ambiente Local com Docker Compose

- Arquitetura multi-serviço funcional
- Serviços sobem com **um único comando**
- Definição de redes, volumes e variáveis de ambiente
- Instruções claras de execução no README

🔹 Conteinerização e Versionamento

- Dockerfiles bem estruturados (uso de multi-stage quando aplicável)
- Versionamento adequado das imagens

ORIENTAÇÕES

Entrega de Trabalho

- Boas práticas de segurança (ex.: usuário não-root, redução de camadas, dependências mínimas)

🔹 **Kubernetes – Produção Mínima**

- Manifests organizados (Deployments, Services)
- Uso de ConfigMaps e Secrets
- Configuração de readiness/liveness probes (quando aplicável)
- Considerações de segurança (ex.: Pod Security Admission ou equivalente), devidamente justificadas

🔹 **CI/CD**

- Pipeline automatizado executando:
  - Build
  - Testes
  - Publicação de artefatos/imagens
- Uso correto de secrets no pipeline
- Validações básicas (lint, testes, scan opcional)

🔹 **Observabilidade, Deploy e Escala**

- Proposta de métricas, logs e traces
- Estratégia de tracing distribuído (conceitual ou instrumentada)
- Definição e justificativa da estratégia de deploy:
  - Rolling, Blue-Green ou Canary
- Estratégia de escalabilidade:
  - HPA e/ou VPA, alinhada aos requisitos do sistema

Vídeo Pitch (até 4 minutos) – 2,0 pontos

- Visão geral da arquitetura do projeto
- Demonstração do ambiente funcionando (local ou conceitual)
- Explicação das decisões técnicas:
  - Containers
  - Kubernetes
  - Pipeline CI/CD
- Estratégia de deploy, observabilidade e escalabilidade

📌 **Publicar no YouTube** (pode ser não listado) e incluir o link no README e no PDF.

ORIENTAÇÕES

Entrega de Trabalho

Roteiro do Estudante

1. Leia o desafio:

Sua primeira tarefa é entender o desafio proposto. Queremos que você compreenda e explore o problema a fundo. Muita atenção para não perder o foco durante o estudo. Você precisa compreender qual é o desafio para não perder isso de vista durante todo o processo.

2. Fontes de Pesquisa:

Este é o momento de pesquisar o que já existe no mercado e ler todas as indicações que o professor fizer. Afinal, antes de pensar em resolver o desafio, é preciso reunir as ferramentas necessárias e reconhecer o que já existe no mercado de trabalho da sua profissão para lidar com esse tipo de situação. Não esqueça de trazer um exemplo concreto de uma solução já existente.

3. Entrega:

Agora é o momento de se tornar um(a) solucionador(a) de problemas. Com base no que você pesquisou, analisou e desenvolveu, faça a entrega do seu trabalho de acordo com o formato solicitado.