# Vídeo 3 – Estágios DevOps

## Parte – Planejamento (Plan)

Pontos-chave:

* - Falta de alinhamento inicial causa retrabalho.
* - Backlog priorizado e critérios de aceite claros.
* - Uso de ágil para cadência e visibilidade.

Diagrama: `diagramas/v3\_p1\_plan.puml`

### Texto de fala:

Um planejamento nebuloso é convite ao retrabalho e às expectativas desalinhadas. Sem prioridades e critérios de aceite, surgem mudanças de última hora que desorganizam todo o ciclo.

Ao unir Dev, Ops e negócio, definimos metas realistas e fatiamos entregas. Kanban/Scrum dão transparência e permitem adaptação rápida.

## Parte – Codificação (Code)

Pontos-chave:

* - Boas práticas, padrões e revisão de código.
* - Versionamento e trunk‑based/feature flags.
* - Documentação colaborativa.

Diagrama: `diagramas/v3\_p2\_code.puml`

### Texto de fala:

Sem padrões e versionamento, a manutenção se torna lenta e arriscada. Revisões estruturadas elevam a qualidade e disseminam conhecimento.

Trunk‑based + feature flags reduzem branches longas e aceleram integração. Documentamos decisões para dar contexto ao time.

## Parte – Construção (Build)

Pontos-chave:

* - Build automatizado e reprodutível.
* - Gestão de dependências (Maven/Gradle).
* - Artefatos versionados (registry).

Diagrama: `diagramas/v3\_p3\_build.puml`

### Texto de fala:

Build manual é fonte de erro. Automatizar garante consistência e velocidade, além de facilitar auditoria.

Registries centralizam artefatos e permitem rastrear exatamente o que foi para produção.

## Parte – Teste (Test)

Pontos-chave:

* - Pirâmide de testes: unit, integration, e2e.
* - Testes de infraestrutura (TestInfra).
* - Qualidade como gate no pipeline.

Diagrama: `diagramas/v3\_p4\_test.puml`

### Texto de fala:

Testes no fim são caros. Validar cedo encurta feedback e evita regressões.

Gates de qualidade param problemas antes de afetarem usuários e mantêm o pipeline confiável.

## Parte – Lançamento (Release)

Pontos-chave:

* - Releases controlados por pipeline.
* - Aprovação automatizada/humana conforme risco.
* - Trilhas por ambiente (dev/stage/prod).

Diagrama: `diagramas/v3\_p5\_release.puml`

### Texto de fala:

Releases manuais concentram risco. Em pipelines, cada etapa é auditável e replicável.

Aprovações e trilhas por ambiente dão segurança e permitem cadência sem sacrificar controle.

## Parte – Implantação (Deploy)

Pontos-chave:

* - Deploy automatizado (GitOps/ArgoCD/Flux).
* - Estratégias: blue‑green, canary, rolling.
* - Infra agnóstica via contêineres.

Diagrama: `diagramas/v3\_p6\_deploy.puml`

### Texto de fala:

Automatizar o deploy elimina passos manuais e acelera MTTR.

Estratégias progressivas diminuem o blast radius e habilitam rollback rápido.

## Parte – Operação (Operate)

Pontos-chave:

* - Alta disponibilidade e auto‑recuperação.
* - Escalonamento horizontal preferencial.
* - Automação de rotina (runbooks).

Diagrama: `diagramas/v3\_p7\_operate.puml`

### Texto de fala:

Operar bem é responder rápido a variações de carga e falhas. Orquestradores ajudam com reschedules e probes.

Runbooks e automações reduzem tempo de resposta e padronizam correções.

## Parte – Monitoração (Monitor)

Pontos-chave:

* - Observabilidade full‑stack (metrics, logs, traces).
* - Alertas baseados em SLOs/SLIs.
* - Feedback para o planejamento.

Diagrama: `diagramas/v3\_p8\_monitor.puml`

### Texto de fala:

Sem visibilidade, não há melhoria contínua. Observabilidade aponta gargalos e antecipa incidentes.

Fechamos o ciclo: aprendizados alimentam o backlog e melhoram arquitetura e práticas.