Sumário

1. Problemas AS-IS que atacaria primeiro 1
2. Como seria a POC da Esteira de DevSecOps 1
2.1 Cenário 1
3. Como a POC vira padrão 2
3.1 Cenário 2
4. Declaração de uso de IA 2
5. VSM 2
6. KT + Templates/Guidelines 3
6. Considerações Finais 3

---

1. Problemas AS-IS que atacaria primeiro

- Ausência de gates automáticos (qualidade e segurança) antes do merge e do deploy.
- Deploy manual e pouco reprodutível (alto risco operacional, rollback lento).
- Segurança reativa (scan tardio), pouca evidência/auditoria e rastreabilidade de artefatos.

---

2. Como seria a POC da Esteira de DevSecOps

2.1 Cenário

Selecionar 1 aplicação de teste backend de preferência (1 serviço) e padronizar o fluxo: PR -> CI (build/test/scan) -> imagem assinada -> CD GitOps -> staging com testes -> gate ->

produção.

- PR: checks obrigatórios (lint, unit test, cobertura mínima, SAST/SCA/Secrets/IaC).

- CI: gerar SBOM, escanear imagem, assinar e publicar no registry com tag imutável.

- CD: deploy via GitOps (Argo CD na minha experiencia); políticas no cluster (Openshift ou AKS nas minhas experiencias) bloqueando imagens sem assinatura/scan.

- Métricas: lead time, change failure rate, MTTR, SLA de vulnerabilidades, policy pass rate.

---

3. Como a POC vira padrão

3.1 Cenário

Pipeline como template reutilizável (YAML parametrizado) + catálogo de políticas (policy-as-code).

- Definition of Done mínima para onboard (sempre coleto do Tech lead o que se espera dos Stakeholders das áreas de negócio) e checklist de exceções.

- Pacote de onboarding (repo exemplo + documentação + KT) e provisionamento via IaC

---

4. Declaração de uso de IA

- Utilizei IA como apoio para organizar o raciocínio, reorganizar uma estrutura de esteira que já implantei em um cliente da consultoria que atuo, revisando os gates e clareza do texto e no diagrama que acompanha este documento como imagem (posso fornecer o arquivo em draw.io se precisarem).
- As decisões técnicas e a adequação ao cenário foram validadas e ajustadas por mim, visto que o modelo, diagrama apresentado foram um case de um projeto de implementação de Gitops com o GOGS (repositório GIT) sendo a fonte de verdade com integração entre Tekton (Openshift Pipelines) fazendo o CI e o ArgoCD fazendo o deploy.
- A camada de CI existia os gates de cobertura e qualidade de código, build das imagens e geração dos artefatos.

---

5. VSM

AS-IS

- Espera/fila para build e deploy (muito manual).
- Testes e segurança no fim do ciclo; defeitos sobem de fase e impactam ambiente de homologação.
- Evidências/auditoria fracas; correções reativas.

TO-BE (POC)

- CI on-demand + gates automáticos no PR.
- Shift-left: SAST/SCA/IaC/Secrets + SBOM + assinatura no CI.
- CD GitOps + deploy progressivo + rollback rápido.
- SLIs/SLOs e feedback para backlog.

Métricas da POC: Lead time, change failure rate, MTTR, % cobertura, vulnerabilidade fora do SLA, policy pass rate.

---

6. KT + Templates/Guidelines

KT em 3 sessões (1h cada)

- Pipeline: visão geral, gates, como debugar falhas e operar correções.
- Segurança: SAST/SCA/IaC/Secrets, SBOM, assinatura, políticas e SLA de correção.
- CD/Operação: GitOps, deploy progressivo, rollback, SLIs/SLOs, runbooks.

Templates e guidelines

- Template de pipeline (YAML) parametrizado por stack e tipo (API/job).
- Chart/manifest base (Helm/Kustomize) com probes, requests/limits, labels, annotations e padrões de logging.
- Policy-as-code (OPA/Gatekeeper ou Kyverno) com baseline (assinatura obrigatória, sem latest, sem privileged, etc.).
- Checklist de PR/release/hotfix/rollback + Definition of Done.

---

6. Considerações Finais

O padrão deste documento eu geralmente uso para estruturar outros modelos de documentação de entrega em projeto como AS-IS, AS-Built (acaba sendo um documento vivo porque vou atualizando conforme vou evoluindo das etapas da implantação) em formato PDF.

O material do KT também entrego em formato PDF com a descrição do conteúdo que foi apresentado e gravação das sessões salvas no repositório ou em página do Confluence caso o cliente tenha.