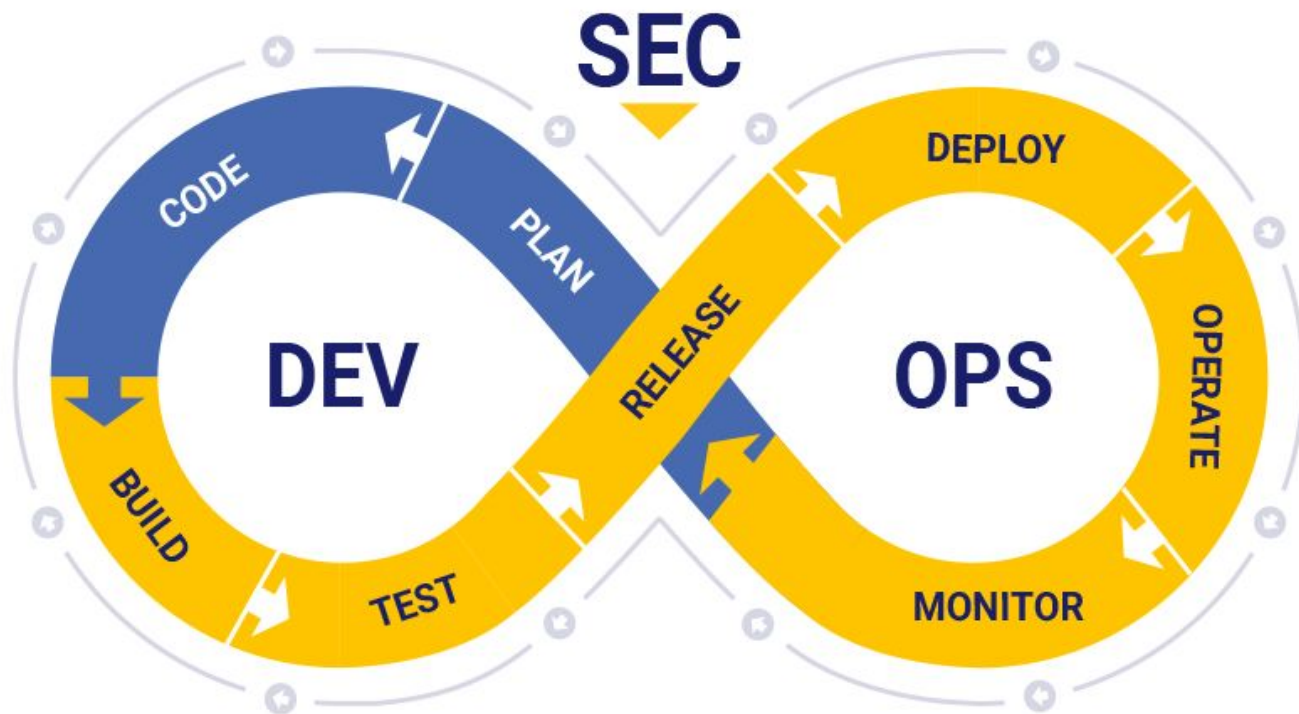


DevSecOps - SAST, DAST

Desenvolvimento, Segurança e Operações

Adicionando segurança em toda a etapa de desenvolvimento.
Professor: Rafael Alexandre Piemontez

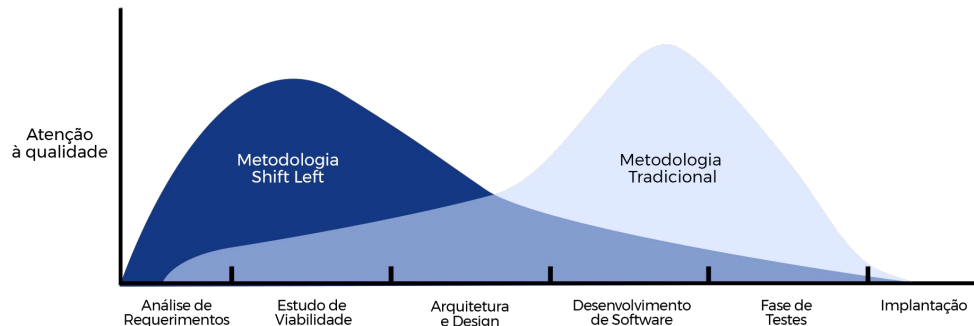
DevSecOps



Shift Left

Em DevSecOps, "shift left" significa integrar a segurança desde as fases iniciais do desenvolvimento de software, em vez de tratá-la apenas no final.

Isso envolve aplicar medidas de segurança em paralelo com as atividades de desenvolvimento, como testes e análise de código, permitindo que problemas de segurança sejam identificados e corrigidos mais cedo, reduzindo o custo e o esforço da correção.



SAST: Static Application Security Testing

SAST (Static Application Security Testing) é uma metodologia de teste de segurança que analisa o código-fonte, bytecode ou binários de uma aplicação sem executá-la.

Ferramentas SAST "lêem" o código-fonte em busca de padrões de vulnerabilidade conhecidos, falhas de configuração, e outras fraquezas que possam levar a problemas de segurança.

Algumas Vantagens:

- **Identificação Precoce:** Encontra vulnerabilidades nas fases iniciais do desenvolvimento, onde são mais baratas e fáceis de corrigir.
- **Visibilidade Profunda:** Oferece uma visão detalhada do código, apontando a linha exata onde a vulnerabilidade reside.
- **Ajuda no Design Seguro:** Fornece feedback aos desenvolvedores para que escrevam código mais seguro desde o início.

Exemplos de Problemas Detectados: Injeção SQL, Cross-Site Scripting (XSS), falhas de autenticação, uso de componentes vulneráveis, erros de configuração de segurança.

SAST no Fluxo DevSecOps

"Shift Left": SAST é um pilar fundamental do conceito "Shift Left" em DevSecOps. Isso significa mover a segurança para as primeiras etapas do SDLC (Software Development Life Cycle), tornando-a uma responsabilidade compartilhada entre desenvolvedores e equipes de segurança.

Integração no Pipeline CI/CD:

- **Antes do Commit (IDE):** Desenvolvedores recebem feedback instantâneo sobre vulnerabilidades enquanto escrevem o código, evitando que falhas cheguem ao repositório.
- **No Build/Teste:** Scans SAST automatizados são executados a cada commit ou pull request, bloqueando a integração de código com vulnerabilidades críticas.
- **Feedback Rápido:** Notificações automatizadas para os desenvolvedores com detalhes das vulnerabilidades e sugestões de correção.

Limitações (Importante Mencionar):

- **Falsos Positivos:** Podem gerar avisos que não são vulnerabilidades reais, exigindo triagem.
- **Não Detecta Problemas em Tempo de Execução:** Não consegue identificar vulnerabilidades que só aparecem quando o código está em execução (ex: falhas de lógica de negócio, problemas de infraestrutura). Para isso, outras ferramentas como DAST são necessárias.

DAST: Dynamic Application Security Testing

DAST (Dynamic Application Security Testing) é uma metodologia de teste de segurança que analisa uma aplicação em tempo de execução, simulando ataques reais contra a aplicação. Ele age como um "usuário malicioso" automatizado.

É tipicamente executado nas fases de teste e homologação, em ambientes de QA, Staging ou até mesmo Produção.

Ferramentas DAST interagem com a aplicação através de sua interface de usuário ou APIs (HTTP/HTTPS), enviando requisições, manipulando entradas e observando as respostas para identificar vulnerabilidades. Elas não precisam de acesso ao código-fonte.

Algumas Vantagens:

- **Visão "Black-Box":** Testa a aplicação como um atacante faria, sem conhecimento interno do código.
- **Detecção em Tempo de Execução:** Identifica vulnerabilidades que só se manifestam quando a aplicação está rodando (ex: problemas de configuração de ambiente, falhas de lógica de negócio, problemas de autenticação em tempo real).
- **Cobertura Abrangente:** Pode encontrar vulnerabilidades em componentes de terceiros, APIs, e interações entre diferentes partes da aplicação que o SAST pode não ver.
- **Baixos Falsos Positivos:** Geralmente tem uma taxa menor de falsos positivos comparado ao SAST, pois as vulnerabilidades são confirmadas através da execução.

DAST no Fluxo DevSecOps

DAST valida o comportamento da aplicação em um ambiente real.

Integração no Pipeline CI/CD:

- **Pós-Build/Deploy:** Após a aplicação ser construída e implantada em um ambiente de teste (ou staging), os testes DAST automatizados são executados.
- **Validação da Aplicação Final:** Verifica como a aplicação se comporta em um ambiente de execução real, incluindo interações com bancos de dados, outros serviços e configurações de infraestrutura.
- **Testes de Regressão de Segurança:** Garante que novas funcionalidades não introduzam novas vulnerabilidades e que as correções anteriores permaneçam eficazes.

Feedback ao Desenvolvedor (Tardio, mas Crítico): Embora o feedback seja mais tardio que o SAST, as vulnerabilidades detectadas por DAST são frequentemente de alto impacto e precisam ser corrigidas com prioridade. O relatório ajuda a reproduzir o problema.

SCA

A SCA (Análise de Composição de Software) é uma prática no ciclo de vida de desenvolvimento seguro (DevSecOps) que foca em analisar os componentes de software de código aberto (OSS) e de terceiros utilizados em uma aplicação.

Algumas Vantagens:

- **Identificação automática:** Descobre todos os componentes de código aberto e suas dependências diretas e indiretas no seu projeto.
- **Gerenciamento de vulnerabilidades:** Verifica se esses componentes possuem vulnerabilidades de segurança conhecidas.
- **Inventário de software:** Cria um inventário detalhado de todos os componentes utilizados.

Como funciona (de forma simplificada)?

- **Varredura:** A ferramenta SCA examina o código-fonte, manifestos de build (ex: pom.xml, package.json) e binários.
- **Identificação:** Compara os componentes encontrados com bancos de dados de vulnerabilidades (ex: NVD) e informações de licenças.
- **Relatório e Alerta:** Gera relatórios detalhados sobre os componentes, suas vulnerabilidades e licenças, alertando as equipes sobre problemas críticos.
- **Correção:** Orienta as equipes na correção, sugerindo versões mais seguras ou alternativas.

PLAN



CODE



BUILD



TEST



DEPLOY



SAST

SCA

Secret Scanning

DAST

IAST

SAST

- Age analisando o código-fonte, sem ser necessária a execução do código;
- As vulnerabilidades são encontradas antes do desenvolvimento e são menos caras para consertar;
- Não consegue identificar problemas relacionados ao tempo e ao ambiente;

Dast

- Age enquanto o aplicativo está em execução;
- As vulnerabilidades são encontradas após o desenvolvimento;
- Consegue identificar problemas relacionados ao tempo e ao ambiente;
- Encontra problemas que não podem ser identificados em uma análise estática.

SonarQube

SonarQube é uma plataforma open-source para inspeção contínua da qualidade e segurança do código.

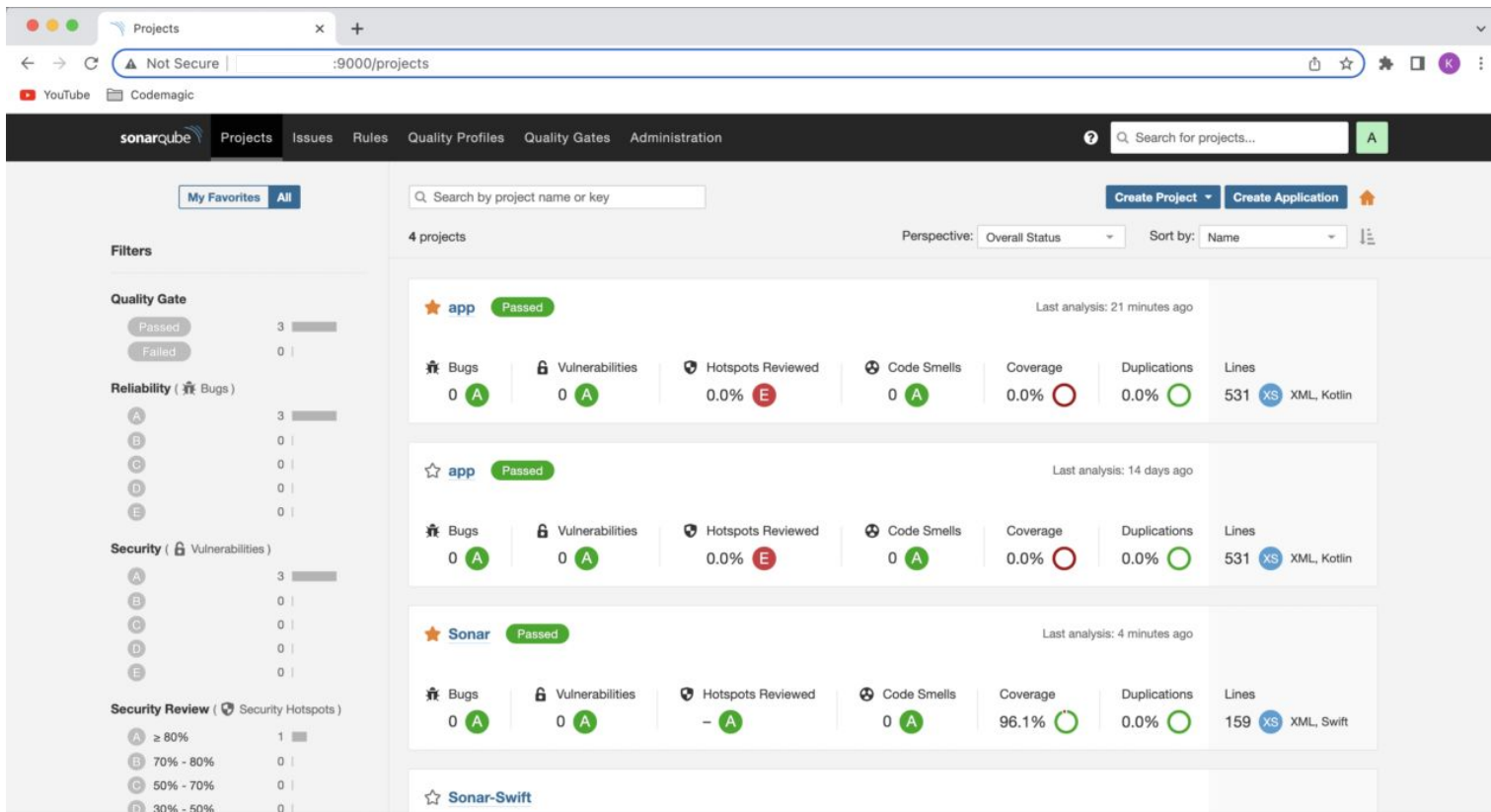
Ele realiza análises estáticas para identificar bugs, vulnerabilidades de segurança, code smells (maus cheiros no código), problemas de cobertura de testes e duplicação de código em diversas linguagens de programação.

O SonarQube analisa o código-fonte (e bytecode) sem executá-lo, aplicando um conjunto extenso de regras e padrões predefinidos (e personalizáveis) para detectar potenciais problemas.

Benefícios:

- **Detecção Precoce de Problemas:** Identifica falhas antes que cheguem à produção, reduzindo custos de correção e riscos.
- **Melhora a Qualidade do Código:** Promove a adoção de boas práticas de programação e facilita a refatoração.
- **Fortalece a Segurança:** Detecta vulnerabilidades comuns e ajuda a prevenir ataques.
- **Aumenta a Eficiência:** Automatiza a revisão de código, liberando tempo dos desenvolvedores para tarefas mais complexas.
- **Visibilidade e Métricas:** Fornece painéis e relatórios com métricas claras sobre a saúde do código.

SonarQube



GitHub Action - Configurar SonarQube

Para integrar a pipeline do Github Action, com o SonarQube, as seguintes ações são necessárias:

- Cadastrar o projeto no sonarqube;
- Coletar o secrete do projeto, no SonarQube;
- Cadastrar o secret no Github Actions;
- Adicionar a etapa do SonarQube;
- Configurar o sonarqube no projeto;

GitHub Action - Segredos - Pasta projeto

Configure o segredo SONAR_TOKEN, com a informação abaixo.

a6981b55326016afac6e4673392b33605b768adb

[Actions secrets](#) / Update secret

SONAR_TOKEN

Value

a6981b55326016afac6e4673392b33605b768adb

GitHub Action - Validação SonarQube Pipeline

No arquivo **.github/workflows/ci.yml**, adicione o bloco sonarqube após build_front

```
build_front:
```

```
...
```

```
sonarqube:
```

```
  name: SonarQube
```

```
  needs: [build_front, teste_back]
```

```
  runs-on: ubuntu-latest
```

```
  steps:
```

```
    - uses: actions/checkout@v4
```

```
      with:
```

```
        fetch-depth: 0
```

```
    - name: SonarQube Scan
```

```
      uses: SonarSource/sonarqube-scan-action@v5
```

```
      env:
```

```
        SONAR_TOKEN: ${ secrets.SONAR_TOKEN }
```

```
empacotar:
```

```
  name: Empacotar
```

```
  needs: [GerarVersao, sonarqube]
```

```
...
```

O código a esquerda, envia o código fonte para a nuvem do sonarqube e o analisa.

GitHub Action - Configurar SonarQube

Crie o arquivo **sonar-project.properties**, na pasta raiz do projeto.

```
sonar.projectKey=aulas_aulas-de-devsecops  
sonar.organization=aulas
```

This is the name and version displayed in the SonarCloud UI.

```
sonar.projectName=Aulas de DevSecOps  
sonar.projectVersion=1.0
```

Path is relative to the sonar-project.properties file. Replace "\" by "/" on Windows.

```
sonar.sources=ci_cd_projeto
```

Encoding of the source code. Default is default system encoding

```
sonar.sourceEncoding=UTF-8
```


GitHub Action


Após configurado o projeto, dispare a pipeline. A pipeline deve exibir o seguinte resultado.







ci.yml

on: workflow_dispatch



SonarQube - Projetos

My Projects My Issues Explore



Filters


Quality Gate

✓ Passed	0
✗ Failed	0

Reliability







A	1
B	0
C	0
D	0
E	0

Perspective Overall Status Sort by Name 1 projects

 [Aulas / Aulas de DevSecOps](#) New Public

Not computed

Last analysis: 5/30/2025, 6:54 PM • 260 Lines of Code • CSS, JavaScript, ...

 0	 1	 1	 100%	 0.0%	 0.0%
Security	Reliability	Maintainability	Hotspots Reviewed	Coverage	Duplications

1 of 1 shown

SonarQube - Endereço Projeto

Endereço do painel com o resultado do SonarQube:

https://sonarcloud.io/project/overview?id=aulas_aulas-de-devsecops

Endereço painel da organização:

<https://sonarcloud.io/organizations/aulas/projects>

SonarQube - Resumo

The screenshot displays the SonarQube web interface. At the top, the navigation bar includes the SonarQube logo, 'cloud' text, and links for 'My Projects', 'My Issues', and 'Explore'. A purple 'Upgrade' button is on the right. Below the navigation bar, the breadcrumb trail shows 'Aulas > Aulas de DevSecOps > master'. The 'Summary' tab is selected, with other tabs like 'Issues', 'Security Hotspots', 'Measures', 'Code', and 'Activity' visible. The main content area is titled 'Main Branch Summary' and shows '260 Lines of Code', 'Version 1.0', and 'Last analysis 1 day ago'. A 'Quality Gate' section indicates 'Not computed' with a minus sign icon. Below this, a table of metrics is shown:

Security	Reliability	Maintainability
0 Open issues A	1 Open issues A	1 Open issues A
Accepted Issues 0 🕒	Coverage 0.0% 🔴 No conditions set on 24 Lines to cover	Duplications 0.0% 🟢 No conditions set on 9.4k Lines
Security Hotspots 0		

SonarQube - Código a revisar

The screenshot displays the SonarQube web interface. At the top, the navigation bar includes the SonarQube logo, 'cloud' text, and links for 'My Projects', 'My Issues', and 'Explore'. A purple 'Upgrade' button is visible on the right. Below the navigation bar, the breadcrumb trail reads 'Aulas > Aulas de DevSecOps > master'. The main content area is divided into a left sidebar and a main panel. The sidebar contains a 'Filters' section with a 'Clear filters' button. Under 'Software Quality', 'Reliability' is selected with a count of 1. Under 'Severity', 'Low' is selected with a count of 1. Under 'Clean Code Attribute', 'Adaptability' is selected with a count of 1. The main panel shows a list of issues. The first issue is 'Do not import modules using an absolute path' located at 'ci_cd_projeto/frontend/src/App.jsx'. It is categorized as 'Reliability' (yellow icon) and 'Maintainability' (red icon). The issue is marked as 'Open' and has a severity of 'L3'. The effort is '5min effort' and it was created '14 days ago'. The issue is labeled as a 'Code Smell' and is 'Critical'. The interface also shows '1 of 1 shown'.

SonarQube cloud My Projects My Issues Explore Upgrade

Aulas > Aulas de DevSecOps > master

Summary Issues Security Hotspots Measures Code Activity

Filters Clear filters X

Software Quality 1 X

Security 0

Reliability 1

Maintainability 1

Add to selection Ctrl + click

Severity ?

Blocker 0

High 0

Medium 0

Low 1

Info 0

Clean Code Attribute

Consistency 0

Intentionality 0

Adaptability 1

Responsibility 0

Bulk Change Select issues Navigate to issue 1 issues 5min effort

ci_cd_projeto/frontend/src/App.jsx

Do not import modules using an absolute path Adaptability

Reliability Maintainability paths pitfall +

Open Rafael Alexandre Piemontez

L3 5min effort 14 days ago Code Smell Critical

1 of 1 shown

SonarQube - Cobertura de testes

The screenshot displays the SonarQube web interface for a project named 'Aulas de DevSecOps'. The top navigation bar includes the SonarQube logo, 'My Projects', 'My Issues', and 'Explore' links. A purple 'Upgrade' button is visible in the top right corner. The sidebar on the left contains icons for project overview, issues, and other metrics. The main content area shows the 'Measures' tab selected, displaying a table of test coverage for 7 files. The table columns are 'Coverage', 'Uncovered Lines', and 'Uncovered Conditions'. All files show 0.0% coverage.

SonarQube cloud My Projects My Issues Explore Upgrade

Aulas > Aulas de DevSecOps > master

Summary Issues Security Hotspots **Measures** Code Activity

Project Overview

Security ? >

Reliability ? >

Maintainability ? >

Security Review ? >

Coverage ▾

Overview

Overall Code

Coverage 0.0%

Lines to Cover 24

Uncovered Lines 24

Line Coverage 0.0%

Aulas de DevSecOps View as List 7 files

Coverage 0.0% See history

	Coverage	Uncovered Lines	Uncovered Conditions
ci_cd_projeto/backend/src/app.controller.spec.ts	0.0%	8	-
ci_cd_projeto/backend/src/app.controller.ts	0.0%	1	-
ci_cd_projeto/backend/test/app.e2e-spec.ts	0.0%	8	-
ci_cd_projeto/frontend/src/App.jsx	0.0%	2	-
ci_cd_projeto/backend/src/app.service.ts	0.0%	1	-
ci_cd_projeto/frontend/src/main.jsx	0.0%	1	-
ci_cd_projeto/backend/src/main.ts	0.0%	3	-

7 of 7 shown

SonarQube - Incluir cobertura de testes

Casa tecnologia (Java, node, php, python,...) possui uma forma diferente de configurar a cobertura de código no Sonar.

Para o projeto (trabalho em aula) node, basta incluir o código, abaixo, ao final do arquivo **sonar-project.properties**.

Linguagens (opcional, mas boa prática para projetos com várias linguagens)

sonar.language=**ts**

Cobertura de Código - Relatórios LCOV

O NestJS com Jest gera lcov.info. SonarQube pode processar múltiplos.

Certifique-se que o caminho esteja correto para onde seus relatórios de cobertura são gerados.

sonar.javascript.lcov.reportPaths=**ci_cd_projeto/backend/coverage/lcov.info**

SonarQube - Remover arquivos da cobertura

Muitos dos arquivos do projeto não precisam ser testados, como por exemplo os próprios arquivos de teste. Para remover estes arquivos informe a propriedade **sonar.exclusions**, ao final do arquivo **sonar-project.properties**.


```
# Arquivos que não precisam ser testados  
sonar.exclusions=**/*spec.ts,**/*.module.ts
```


SonarQube - Resultado esperado

Search for projects Perspective Overall Status Sort by Name 1 projects

★ Aulas / Aulas de DevSecOps New Public ✓ Passed

Last analysis: 6/1/2025, 3:50 PM • 213 Lines of Code • CSS, JavaScript...

A 0	A 1	A 1	A 100%	 52.9%	0.0%
Security	Reliability	Maintainability	Hotspots Reviewed	Coverage	Duplications

1 of 1 shown

Referências

Gene Kim; **Manual de DevOps: Como Obter Agilidade, Confiabilidade e Segurança em Organizações Tecnológicas**; Alta Books; ISBN 978-8550802695

Steve Suehring; **Learning DevSecOps: A Practical Guide to Processes and Tools**; O'Reilly Media; ISBN: 9781098144869