Proposta de uma arquitetura DevSecOps e os impactos nos controles do PPSI (Programa de Privacidade e Segurança da Informação): Um estudo de caso

Tharcísio Mendonça
PPEE
Universidade de Brasília - UNB
Brasília, Brasil
tharcisio.mendonca@aluno.unb.br

Prof. Drº Robson de Oliveira Albuquerque
PPEE
Universidade de Brasília - UNB
Brasília, Brasil
robson@redes.unb.br

***Resumo*—Este artigo propõe uma arquitetura automatizada de DevSecOps em ambiente de nuvem, a partir de um estudo de caso, com o objetivo de avaliar seus impactos sobre os controles do Programa de Privacidade e Segurança da Informação (PPSI) e as boas práticas do CIS Controls v8. De natureza aplicada e exploratória, a pesquisa foi validada por meio de um laboratório experimental que integrou pipelines de integração e entrega contínuas (CI/CD) com GitHub Actions, Docker, Kubernetes, SonarQube, Trivy, Zaproxy, Prometheus, Grafana e Trend Micro Workload Security, em infraestrutura provisionada na nuvem da DigitalOcean. A arquitetura promove a incorporação sistemática da segurança desde as fases iniciais do ciclo de vida do software, assegurando automação, rastreabilidade e governança contínua. Os resultados demonstram ganhos significativos em padronização de ambientes, qualidade e segurança do código, eficiência operacional e observabilidade, além de evidenciar a contribuição da arquitetura para o fortalecimento da maturidade em DevSecOps e para a conformidade institucional com o PPSI e o CIS Controls v8. A pesquisa oferece uma arquitetura replicável voltado a instituições públicas de ensino, pesquisa e saúde, contribuindo para o avanço da governança digital e da resiliência cibernética.**

***Index Terms*—DevSecOps, Segurança da Informação, PPSI, CIS Controls v8, CI/CD**

I. Introdução

As instituições públicas de ciência e tecnologia em saúde exercem papel estratégico na consolidação do Sistema Único de Saúde (SUS) e na promoção da inovação científica e tecnológica [1]. A crescente digitalização de processos e serviços intensificou a dependência de infraestruturas em nuvem e aplicações web, ampliando o volume de dados sensíveis sob gestão. Nesse cenário, a proteção da informação e a conformidade com políticas de segurança tornaram-se requisitos fundamentais para assegurar a continuidade dos serviços e a confiança da sociedade. Entretanto, observa-se que a maturidade em segurança digital nessas instituições ainda é incipiente, marcada por processos fragmentados e baixa automação, o que limita a eficiência operacional e a capacidade de resposta a incidentes.

Nas últimas décadas, o avanço das metodologias ágeis e o uso de infraestrutura em nuvem transformaram profundamente o ciclo de desenvolvimento de software, exigindo maior integração entre equipes de desenvolvimento e operação. O movimento DevOps emergiu como resposta a essa demanda [2], promovendo automação, colaboração e entrega contínua. Contudo, a ausência de práticas de segurança integradas desde as fases iniciais do ciclo de vida do software (SDLC) revelou limitações significativas em termos de confiabilidade e conformidade [3]. Para mitigar essas lacunas, a literatura recente tem enfatizado a evolução do DevOps para o DevSecOps, que incorpora segurança como elemento transversal e automatizado em todas as etapas do pipeline [4].

Estudos como os de Constante [4], Thota [5] e Alghawli e Radivilova [6] evidenciam o potencial da automação de testes de segurança, da análise estática e dinâmica de código, e da integração de infraestrutura como código (IaC) em ambientes de CI/CD. Outros trabalhos, como Rangaraju et al [7] e Verdet e Silva [8], destacam o papel emergente da inteligência artificial e da observabilidade contínua na detecção de ameaças e no reforço da conformidade. Em paralelo, frameworks de referência, como o CIS Controls v8 e o OWASP DevSecOps Maturity Model (DSOMM), têm orientado a implementação de controles técnicos e processuais voltados à maturidade em segurança. No contexto governamental brasileiro, o Programa de Privacidade e Segurança da Informação (PPSI) tornou-se um instrumento essencial para alinhar práticas institucionais às exigências da Lei Geral de Proteção de Dados (LGPD) e às boas práticas internacionais de governança da informação.

Apesar dos avanços observados, a literatura ainda carece de estudos aplicados que avaliem como a adoção de arquiteturas automatizadas de DevSecOps impacta diretamente os controles de segurança e privacidade em instituições públicas, especialmente em ambientes de nuvem privada. A maior parte das abordagens documentadas concentra-se em contextos corporativos ou industriais, com foco em padronização de pipelines e automação de verificações técnicas, mas sem examinar sua

relação com programas de conformidade e governança da informação. Além disso, observa-se uma lacuna na integração entre mecanismos técnicos de segurança contínua e diretrizes institucionais de privacidade e proteção de dados, o que limita a capacidade de organizações públicas evoluírem para níveis mais altos de maturidade em DevSecOps. Há, portanto, uma necessidade de estudos que correlacionem práticas técnicas automatizadas com frameworks normativos, fornecendo evidências sobre os benefícios e desafios dessa integração.

Diante dessas lacunas, este trabalho tem como objetivo propor uma arquitetura automatizada de DevSecOps em nuvem, capaz de integrar segurança, privacidade e governança ao ciclo de vida de desenvolvimento de software, avaliando seus impactos sobre os controles do PPSI e as boas práticas do CIS Controls v8. A pesquisa adota uma abordagem aplicada e exploratória, apoiada na construção de um laboratório experimental que implementa pipelines de integração e entrega contínuas com GitHub Actions, Docker, Kubernetes, SonarQube e PostgreSQL.A proposta contribui para o avanço do conhecimento científico e aplicado em segurança no contexto DevSecOps, ao apresentar uma arquitetura replicável destinada a instituições públicas que buscam fortalecer sua maturidade tecnológica e postura de segurança da informação. O estudo baseia-se em um caso prático conduzido em um órgão público voltado ao ensino, à pesquisa e à saúde pública, configurando uma referência para a adoção de práticas integradas de segurança, automação e governança digital.

O artigo está estruturado da seguinte forma: a Seção II descreve a metodologia e o delineamento experimental; a Seção III revisa os trabalhos relacionados; a Seção IV apresenta o estudo de caso e a arquitetura proposta; a Seção V analisa os resultados obtidos; e a Seção VI discute as conclusões e direções para trabalhos futuros.

II. Metodologia

No contexto científico, a pesquisa pode ser classificada de acordo com diferentes critérios. Entre eles, é possível diferenciar tipos de pesquisa de acordo com sua natureza, objetivos ou procedimentos técnicos [9].

A presente pesquisa adota natureza aplicada, pois visa gerar conhecimento direcionado à solução de problemas práticos [9], especificamente o desenvolvimento de uma arquitetura automatizada de DevSecOps em uma instituição pública de ensino, pesquisa e saúde.

Quanto aos objetivos, classifica-se como exploratória, uma vez que busca compreender e avaliar os impactos da adoção do DevSecOps nos controles do Programa de Privacidade e Segurança da Informação (PPSI), proporcionando uma visão inicial e abrangente sobre o tema.

Em relação à estratégia metodológica, a pesquisa é bibliográfica, documental e de estudo de caso.A pesquisa bibliográfica envolveu a análise de artigos, teses, livros e publicações indexadas, com buscas conduzidas em bases como Scopus, IEEE Xplore, Web of Science, ResearchGate e ACM Digital Library. Foram utilizados os termos *"devsecops and cloud"* e *"devops and cloud"*, aplicando-se critérios de inclusão e exclusão, priorizando trabalhos mais citados, com maior fator de impacto e mais recentes.

A pesquisa documental apoiou-se em normas, políticas e controles relacionados ao PPSI e ao CIS Controls v8, fornecendo o arcabouço normativo para análise. O estudo de caso será conduzido em uma instituição pública voltada ao ensino, à pesquisa e à saúde pública, possibilitando a validação da proposta em um ambiente real e multidisciplinar. Ao término, a arquitetura desenvolvida será testada em laboratório, e seus resultados serão analisados quanto aos impactos nos controles do PPSI (Programa de Privacidade e Segurança da Informação).

III. Trabalhos relacionados

Este capítulo apresenta os principais estudos sobre o tema, abordando práticas, ferramentas e modelos que integram segurança ao ciclo de desenvolvimento ao DevSecOps.

A. Etapas DevSecOps

Os aspectos de segurança do DevSecOps abrangem a responsabilidade compartilhada pela segurança, a automação de processos de segurança, a integração contínua de testes de segurança, a implementação de medidas de segurança proativas e a colaboração entre as equipes. O princípio fundamental é a "mudança para a esquerda"("shift left"), que consiste em integrar a segurança nas fases iniciais do SDLC, em vez de considerá-la apenas como uma verificação final [2], [4], [10].

Alghawli e Radivilova desenvolveram um modelo de segurança DevSecOps voltado para um cluster de nuvem resiliente, combinando ferramentas e metodologias automatizadas. A proposta incluiu um algoritmo de avaliação de risco quantitativa em tempo real, baseado na metodologia FAIR, e o uso do Método de Análise Hierárquica (AHP) para selecionar o provedor de nuvem mais adequado entre AWS, Azure e GCP. A arquitetura foi implementada com Infraestrutura como Código (Terraform) e pipeline CI/CD com Jenkins. Os resultados indicaram a AWS como provedora mais adequada, e os testes comprovaram a eficácia do modelo, com cálculo automatizado de risco financeiro (US$ 5.434,54) em menos de um segundo durante a simulação de uma vulnerabilidade em um bucket S3. [6].

O framework OWASP DevSecOps Guideline enfatiza a importância de incorporar a segurança em todas as fases do ciclo de vida do software [11]. Os principais objetivos alcançados com as análises e os testes propostos no modelo DevSecOps são:

- Identificar vulnerabilidades de software precocemente [6], [7], [10]
- Reduzir riscos de segurança e de negócios. [6], [8], [10]
- Acelerar a velocidade do desenvolvimento com feedback oportuno [10]
- Garantir a conformidade regulatória [10]
- Construir uma cultura de segurança [7]
- Melhorar a postura de segurança de aplicativos e infraestrutura [10]

- Diminuir o tempo e o custo de correção de problemas de segurança

O DevSecOps expande os conceitos centrais do DevOps ao incorporar a segurança em cada fase do SDLC [3]. Isso abrange planejamento, codificação, construção, teste, implantação, operação e monitoramento, conforme apresentado na figura 1.

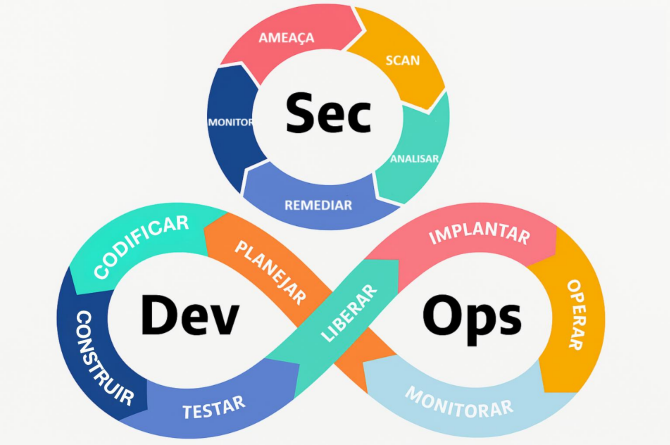

Figura 1. Etapas DevSecOps (adaptado de Pessol [12]

Com base nos trabalhos relacionados [3]–[5], [11], [13]–[15], form mapeadas algumas ferramentas para cada etapa, conforme tabela I:

Tabela I
FASES DO DEVSECOPS E PRINCIPAIS FERRAMENTAS E CONTROLES

Fase do CI/CD	Objetivo Principal	Ferramentas DevSecOps	Tipo de Controle
Planejamento (Plan)	Definir requisitos, riscos e políticas de segurança	Jira, Confluence, ThreatModeler, OWASP Threat Dragon, GitHub Projects	Planejamento de backlog seguro, modelagem de ameaças, definição de políticas e conformidade
Codificação (Code)	Garantir código seguro desde o desenvolvimento	GitHub / GitLab / Bitbucket, Visual Studio Code, GitLeaks, SonarLint, Semgrep	Controle de versão, revisão de código, detecção de segredos, análise estática local
Construção (Build)	Compilar e empacotar com segurança	Jenkins, GitHub Actions, GitLab CI/CD, Trivy Azure DevOps, SonarQube, Snyk, Dependency-Check	CI, análise SAST e SCA, verificação de dependências e containers
Testes (Test)	Validar segurança funcional e lógica da aplicação	OWASP ZAPROXY, Burp Suite, Nikto, Gauntlt, Trivy, Checkov, KubeLinter	DAST – análise dinâmica de aplicações, varredura de vulnerabilidades em execução, testes automatizados de segurança no pipeline
Liberação/ Implantação (Release/ Deploy)	Entregar software seguro e rastreável	ArgoCD, Spinnaker, Helm, Terraform, Ansible, HashiCorp Vault, Cosign	CD seguro, assinatura digital, controle de segredos, infraestrutura como código
Operação (Operate)	Monitorar e manter o ambiente seguro	Prometheus, Grafana, ELK Stack, Splunk, Wazuh, Sysdig Secure, Falco	Observabilidade, monitoramento de logs e alertas, runtime security
Monitoração e Melhoria contínua (Monitor e Feedback)	Aprendizado contínuo e resposta a incidentes	Grafana Loki, Datadog, ELK, OpenTelemetry, Jira Service Management	Métricas, auditorias, incident response, melhoria contínua

B. Padrões de segurança e maturidade em pipelines DevSecOps

Constante et al. propuseram uma abordagem sistemática para integrar requisitos de padrões de segurança em pipelines DevOps, voltada especialmente para ambientes industriais altamente regulados, como os Sistemas de Controle Industrial (ICS) [4]. O estudo aborda o conflito entre conformidade com normas rigorosas, como a IEC 62443-4-1, e a manutenção de curtos tempos de entrega, conforme apresentado na figura. 2. A análise demonstrou que 31% das atividades do padrão podem ser totalmente automatizadas, enquanto 38% ainda exigem intervenção humana. Práticas como testes de segurança (SVV) e gerenciamento de atualizações (SUM) apresentaram alto potencial de automação, ao passo que especificação de requisitos (SR) e criação de diretrizes de segurança (SG) permanecem majoritariamente manuais. A avaliação com profissionais da indústria confirmou a utilidade da arquitetura para construir e avaliar pipelines compatíveis com padrões de segurança.

Pinto propõe o desenvolvimento de um software de apoio à adoção da metodologia DevSecOps, integrando desenvolvimento, segurança e operações de forma automatizada. A solução, fundamentada em revisão de literatura e nas melhores práticas de segurança, incorpora análises SAST, DAST e SCA, além de automação de infraestrutura e versionamento, visando aumentar a transparência, reduzir o tempo de desenvolvimento e aprimorar a qualidade e a segurança das aplicações [3]. Aplicada inicialmente no Tribunal Regional do Trabalho da 21ª Região, a proposta contribui para disseminar a cultura DevSecOps e oferece um modelo replicável para outros órgãos públicos.

C. Integração de Infraestrutura como Código (IaC) em pipelines CI/CD para automação em Nuvem

Thota destaca o desafio de incorporar segurança em ambientes de desenvolvimento nativos da nuvem, que são rápidos e ágeis. A pesquisa utilizou uma abordagem mista, combinando revisão de literatura, análise de casos reais (Google Cloud e Microsoft) e experimentos práticos em um pipeline CI/CD baseado em Kubernetes [5]. Foram integradas ferramentas de SAST, DAST, scanners de dependências e IaC para avaliar sua eficácia. Os resultados mostraram alta precisão na detecção de vulnerabilidades — 92% para SAST e mais de 80% para IaC —, embora com sobrecarga de desempenho de 12–18% no tempo de execução. O estudo também destacou desafios de complexidade das ferramentas, falsos positivos (9%) e resistência cultural dos desenvolvedores, concluindo que a automação de segurança é essencial e transformadora para o DevSecOps, equilibrando velocidade e segurança [5].

Verdet e Silva realizaram um estudo empírico sobre a adoção de práticas de segurança em Infraestrutura como Código (IaC), considerando os riscos de vulnerabilidades introduzidas por configurações incorretas em ferramentas como o Terraform. A pesquisa categorizou 287 políticas de segurança para AWS, Azure e GCP, com base em padrões da indústria, e analisou 812 projetos de código aberto no GitHub utilizando a ferramenta Checkov [8]. Os resultados

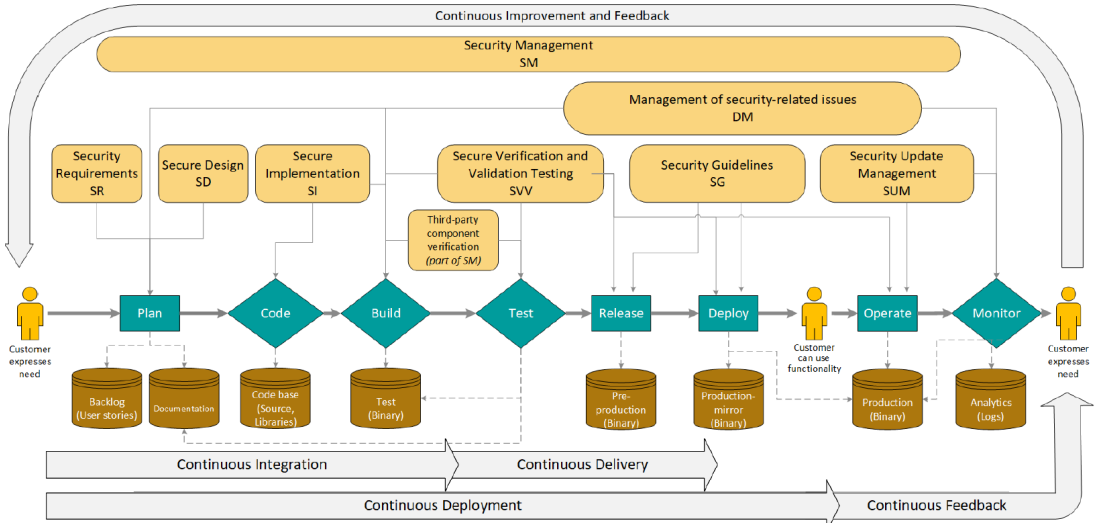

Figura 2. Pipeline DevOps em conformidade com o padrão de segurança IEC 62443-4-1 (tradução nossa) [4]

mostraram que as políticas de controle de acesso são as mais implementadas, enquanto as de criptografia em repouso são frequentemente negligenciadas. Além disso, foi identificada uma correlação positiva entre a popularidade dos repositórios (número de estrelas) e a adoção de práticas de segurança, indicando que projetos mais reconhecidos tendem a manter configurações de infraestrutura mais seguras [8].

Rangaraju et al. investigaram a integração de estratégias de Inteligência Artificial (IA) no framework DevSecOps para fortalecer a segurança em ambientes de nuvem. O estudo destaca o DevSecOps como uma mudança cultural que une desenvolvimento, segurança e operações, enquanto a IA amplia suas capacidades ao possibilitar detecção avançada de ameaças, gestão proativa de riscos e respostas automatizadas [7]. Os autores demonstram como algoritmos de IA podem analisar grandes volumes de dados em tempo real, identificar anomalias e adaptar-se a ameaças emergentes. A pesquisa também detalha a aplicação da IA nas fases de Integração Contínua (CI), Entrega Contínua (CD) e Infraestrutura como Código (IaC), além de discutir desafios técnicos e éticos da implementação, propondo boas práticas e recomendações para uma adoção eficaz.

D. Relação DevSecOps com Framework PPSI – Programa de Privacidade de Segurança da Informação

O Programa de Privacidade e Segurança da Informação (PPSI) é crucial para os órgãos do Governo Federal, visando elevar a maturidade e a resiliência em privacidade e segurança da informação. Ele estabelece diretrizes e controles para proteger dados e sistemas de informação, aumentando a confiança dos cidadãos nos serviços digitais [16]. A não conformidade com o PPSI pode acarretar diversos impactos negativos para a organização, incluindo:

- Sanções financeiras: Multas por descumprimento das regulamentações de privacidade e segurança, como a LGPD, que o PPSI visa apoiar [16]
- Danos à reputação: Perda de confiança dos cidadãos e de outras partes interessadas devido a falhas na proteção de dados e segurança dos sistemas
- Responsabilidade legal: Ações judiciais e outras consequências legais decorrentes de incidentes de segurança e violações de privacidade [2]
- Interrupção de serviços: Incidentes de segurança podem levar à indisponibilidade de sistemas e serviços críticos [8].
- Perda de dados: Violações de segurança podem resultar na perda ou roubo de informações confidenciais [8], [10]
- Escrutínio regulatório: Maior supervisão e auditorias por parte dos órgãos reguladores [2], [10]

A integração de controles de segurança e privacidade nos pipelines reforça a efetividade do PPSI, garantindo conformidade com normas de governança e princípios de confidencialidade, integridade e disponibilidade. Essa abordagem automatiza verificações de compliance, reduz vulnerabilidades, fortalece controles de acesso e aprimora a rastreabilidade de riscos, alinhando-se à LGPD. Assim, as instituições aumentam a eficiência operacional e mantêm o compromisso ético e legal com a proteção de dados sensíveis. Dessa forma, ao adotar esta arquitetura automatizada de DevSecOps em um ambiente de nuvem privada, busca-se obter os seguintes benefícios:

- Melhoria na detecção e resposta a ameaças cibernéticas [17].
- Redução de vulnerabilidades e riscos de segurança [10].
- Aceleração do ciclo de vida do desenvolvimento com segurança [2], [17]
- Conformidade contínua com regulamentações de segurança [2], [10]
- Redução no número de incidentes de segurança [7]

A correlação do DevSecOps com os controles do PPSI pode ser estabelecida conforme tabela II:

Tabela II
RELAÇÃO DO OWASP DEVSECOPS GUIDELINE COM CONTROLES DO PPSI

Framework "OWASP DevSecOps Guideline"	Framework "PPSI"	Correlação entre Frameworks
Análise de Requisitos de Segurança e Modelagem de Ameaças	Control 16: Segurança de Aplicações	A modelagem de ameaças e a análise de requisitos de segurança estão incluídas no controle de segurança de aplicações e nos processos de desenvolvimento seguro
Análise de Código Estático (SAST) e Análise de Composição de Software (SCA)	Control 16: Segurança de Aplicações	Ferramentas de análise estática e dinâmica verificam práticas de codificação seguras e o gerenciamento de componentes de terceiros faz parte da segurança
Varredura de Segredos	Control 3: Proteção de Dados Control 4: Configuração Segura de Ativos Institucionais e Software Control 5: Gestão de Contas	A proteção de dados sensíveis, configuração segura e gestão de contas abordam a varredura de segredos.
Análise de Configuração de IaC	Control 4: Configuração Segura de Ativos Institucionais e Software	A configuração segura de ativos contempla o uso de versionamento e IaC.
Varredura de Contêineres	Control 16: Segurança de Aplicações Control 2: Inventário e Controle de Ativos de Software Control 7: Gestão Contínua de Vulnerabilidades	A segurança de contêineres é tratada por controles de configuração, inventário e gestão de vulnerabilidades.
Testes de Segurança Dinâmicos (DAST)	Control 16: Segurança de Aplicações	Ferramentas dinâmicas são usadas para testes de segurança em aplicações.
Testes de Segurança Interativos (IAST)	Control 16: Segurança de Aplicações	Análise estática e dinâmica é combinada para o ciclo de vida da segurança de softwares.
Testes de Segurança de API	Control 16: Segurança de Aplicações	A segurança de APIs é integrada ao desenvolvimento seguro e à verificação no código
Testes de Penetração	Control 18: Testes de Invasão	Testes de invasão simulam ações de invasores e estão ligados a auditorias.
Testes de Segurança de Tempo de Execução (RASP)	Control 13: Monitoramento e Defesa da Rede Control 16: Segurança de Aplicações	Proteção em tempo de execução está presente no monitoramento de rede e na segurança de aplicações
Auditorias de Segurança	Control 30: Avaliação de Impacto, Monitoramento e Auditoria Control 8: Registros (Logs) de Auditoria Control 18: Testes de Invasão	Auditorias verificam a eficácia das medidas de proteção e são associadas a logs e testes de invasão.
Monitoramento de Segurança	Control 7: Gestão Contínua de Vulnerabilidades Control 13: Monitoramento e Defesa da Rede Control 1: Inventário e Controle de Ativos Institucionais Control 30: Avaliação de Impacto, Monitoramento e Auditoria	Monitoramento envolve defesa de rede, gestão de vulnerabilidades, inventário de ativos e auditoria, incluindo o uso de SIEM
Treinamento e Conscientização em Segurança	Control 14: Conscientização e Treinamento de Competências sobre Segurança Control 23: Conscientização e Treinamento Control 16: Segurança de Aplicações	Controles específicos garantem treinamento e conscientização em cibersegurança, privacidade e codificação segura.

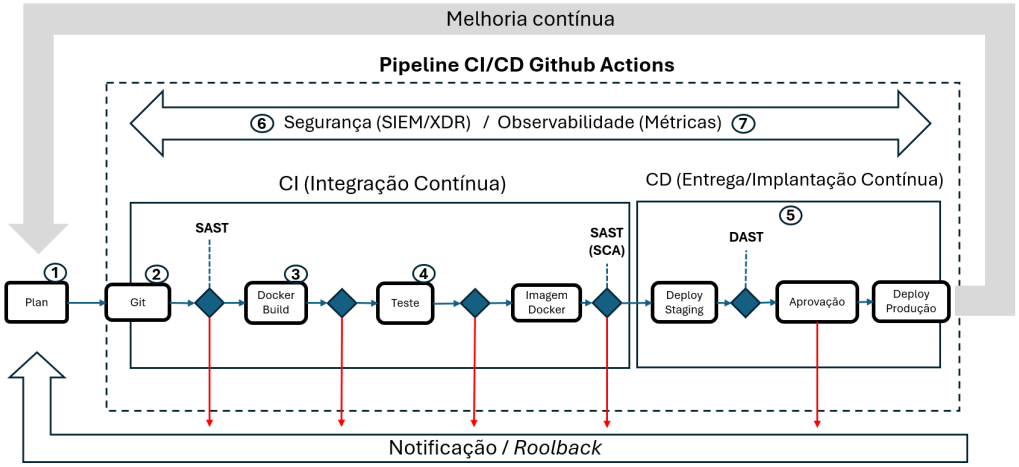

Figura 3. Arquitetura DevSecOps proposta (Elaborado pelo autor)

IV. PROPOSTA DA ARQUITETURA

A proposta apresenta uma arquitetura de pipeline DevSecOps que utiliza tecnologias amplamente adotadas na indústria, como Docker, Kubernetes, GitHub Actions e SonarQube, com o objetivo de incorporar segurança e automação em todas as etapas do processo. Além disso, foram consideradas as soluções tecnológicas já adotadas pela instituição onde o estudo de caso será realizado, de modo a garantir aderência ao ambiente existente e maior viabilidade prática da proposta.

Conforme apresentado na figura 3, a arquitetura está divida em fases e contempla ambientes de homologação (staging) e produção, propondo práticas que vão desde o versionamento seguro do código até o monitoramento contínuo em runtime, a partir de pipelines CI/CD. Ao detalhar cada fase, busca-se demonstrar como a integração de ferramentas e metodologias pode reduzir riscos, aumentar a confiabilidade das entregas e fortalecer a postura de segurança organizacional. A seguir, apresentam-se as fases da arquitetura, acompanhadas das ferramentas utilizadas e das ações previstas em cada etapa:

1) ***Planejamento e Versionamento****:* Definir requisitos, políticas e riscos de segurança antes do desenvolvimento.

Ferramenta:

- GitHub: *Push* e *Pull Request* (PR) na *brach main* (Código para produção)

Ações:

- Definição de estratégia de branching:
  main (produção), develop (homologação), feature/*, hotfix/*.
- Configuração de branch protection rules:
  PR obrigatório, revisão dupla, status checks obrigatórios (build + SonarQube).
- Segurança no repositório:
  Commit signing (GPG) para autenticidade.
  Dependabot para alertas de dependências vulneráveis.
- Integração com Issues/Projects para rastrear backlog de segurança e funcionalidades.
- Criação da Worflow para execução da pipeline via Github Actions

Saídas:

- Código versionado, com baseline de segurança aplicado.
- Requisitos de segurança documentados.
- Backlog priorizado.
- Estrutura de repositório e controle de acesso configurados.

2) ***Codificação (Code)****:* Garantir código limpo, seguro e rastreável desde o início.

Ferramenta:

- GitHub → versionamento, code review e integração com ferramentas de segurança.
- SonarQube (SAST) → análise estática do código, identificando vulnerabilidades, más práticas e falhas de segurança antes da compilação.

Ações:

- Build da aplicação:
  Multi-stage Dockerfile (fase build + fase runtime slim).
  Validação de Dockerfile com Hadolint.
- Testes automatizados:
- Integração → containers efêmeros no GitHub Actions.
- SonarQube Scan:
  Quality gate configurado (mínimo de 80% cobertura de testes, 0 vulnerabilidades críticas).
  Relatórios salvos como artefatos no pipeline.

Saídas:

- Controle de versões rastreável no GitHub.
- Código seguro e validado (sem vulnerabilidades críticas).
- Relatórios SAST (SonarQube).

3) ***Construção (Build)****:* - Criar artefatos de build e imagens de contêiner seguras.

Ferramentas:

- Docker → empacotamento e isolamento de aplicações em contêineres.

Ações:

- Build de imagem com tags:
  app:1.0.0 (semântica) + app:commit-sha + app:latest.
- Políticas de aprovação:
- Publicação no repositório de imagens

Saídas:

- Criação da imagem docker
- Imagem publicada no repositório

4) ***Testes (Test)****:* - Validar a segurança da imagem em execução e do ambiente.

Ferramenta:

- Trivy (SCA) → análise de vulnerabilidades em imagens Docker, bibliotecas e dependências de código aberto.

Ações:

- Scan de imagem Docker com Trivy.
- SCA (dependências):
  Trivy verifica pacotes/bibliotecas vulneráveis (CVEs).

Saídas:

- Imagem Docker validada e livre de vulnerabilidades críticas.
- Relatórios SCA (Trivy).
- Artefatos prontos para teste e implantação.

5) ***Liberação e Implantação (Release/Deploy)****:* Realizar deploy automatizado e seguro da aplicação.

Ferramentas:

- OWASP ZAP (ZAPROXY) (DAST) → executa testes dinâmicos simulando ataques reais contra a aplicação em execução.

Ações (Staging/Homologação):

- Aplicação de manifests Kubernets
- Namespaces Staging
- DAST (Dynamic Application Security Testing): detecta vulnerabilidades como injeções, autenticação fraca e exposição de dados sensíveis.

Aprovação (Manual):

- Responsáveis: Líder de desenvolvimento (Qualidade) e Segurança da Informação
- Relatórios SonarQube.
- Scans de imagem Trivy.
- Logs dos testes DAST/performance.
- Registro formal da aprovação (evidências armazenadas).

Ações (Produção):

- Aplicação de manifests kubernets
- Namespaces Producao
- Observabilidade: Prometheus (métricas), Grafana (dashboards)

Saídas:

- Aplicação implantada com políticas de segurança definidas.
- Infraestrutura como código rastreável.
- Relatórios DAST com vulnerabilidades classificadas.
- Correções priorizadas antes do deploy.
- Aplicação validada em produção.

6) ***Operação (Operate)****:* Monitorar os ambientes, garantir conformidade e detectar incidentes.

Ferramentas: Trend Micro Workload Security – proteção em tempo de execução (runtime security).

Ações:

- Monitorar cargas de trabalho em execução (VMs, containers, pods).
- Detectar vulnerabilidades e comportamentos anômalos.
- Aplicar correções automáticas e políticas de conformidade.

Saídas:

- Logs e alertas de segurança centralizados.
- Ambiente protegido contra ameaças em tempo real.
- Relatórios de conformidade e vulnerabilidades.

*7) **Monitoração e Melhoria contínua**:* Obter visibilidade contínua e promover melhoria do ciclo DevSecOps.

Ferramentas:

- Prometheus – coleta de métricas de desempenho e eventos.
- Grafana – visualização de dashboards e alertas.

Ações:

- Observabilidade completa:
  Logs centralizados
  Métricas de performance e segurança (Prometheus).
  Dashboards em Grafana.
- Alertas automáticos.
- Segurança contínua:
  TrendMicro Workload detecta execuções não autorizadas.
  Revisão periódica de imagens no registry.
- Feedback loop:
  Incidentes e vulnerabilidades viram issues no backlog.
  Reforço da cultura de segurança no ciclo.

Saídas:

- Dashboards e alertas de segurança operacionais.
- Indicadores de desempenho e segurança (KPIs).
- Ações corretivas e melhorias para o ciclo seguinte.

V. Implementação da Arquitetura

Este capítulo apresenta o processo de implementação da arquitetura DevSecOps proposta, tomando como base o cenário real do estudo de caso. Inicialmente, descreve-se o ambiente atual e seus principais desafios; em seguida, realiza-se a avaliação de maturidade utilizando o modelo OWASP DSOMM, identificando lacunas e oportunidades de melhoria. Por fim, detalha-se a implementação prática da arquitetura, demonstrando como suas etapas, ferramentas e controles foram integrados para validar o modelo em ambiente de nuvem.

A. Cenário atual

Uma instituição centraliza sua oferta de tecnologia para suas diversas unidades internas por meio de uma infraestrutura de nuvem privada. Uma parcela significativa dos serviços disponibilizados consiste na provisão e gerenciamento de aplicações e websites. Essa arquitetura é sustentada por um modelo de Infraestrutura como Serviço (IaaS), no qual a equipe de TI dessas unidades é responsável por gerenciar os recursos computacionais subjacentes — como servidores virtuais, armazenamento e redes — sobre os quais as plataformas CMS são instaladas e gerenciadas por essas unidades, semelhante ao apresentado na figura 4.

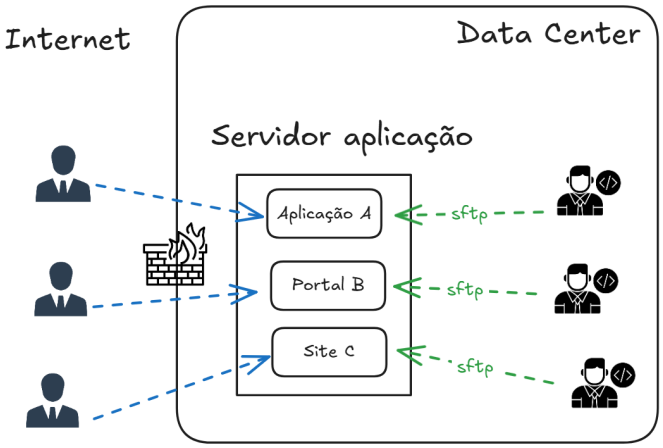

Figura 4. Cenário atual do estudo de caso

Este modelo, embora flexível, apresenta desafios de padronização, segurança e manutenção em escala, uma vez que cada site pode ter suas próprias configurações, plugins e ciclos de atualização. Além do mais, podemos destacar os principais problemas:

1) Cada aplicação precisa ser instalada diretamente no servidor (físico ou virtual).
2) O sistema operacional precisa ter as dependências (bibliotecas, pacotes, frameworks, versões de linguagem) compatíveis com cada aplicação.
3) A configuração é manual e altamente suscetível a erros (ex: conflitos de versões, "funciona na minha máquina mas não no servidor").
4) Escalabilidade é limitada: replicar o ambiente exige reinstalação ou clonagem complexa de servidores.
5) Atualizações e rollback são trabalhosos: é necessário mexer diretamente em pacotes e configurações de produção.
6) Não é escalável → Deploy manual via SFTP não acompanha o crescimento e exige intervenção humana.
7) Não é seguro → Desenvolvedores têm acesso direto à produção; credenciais podem vazar; ausência de DevSecOps (pipeline automatizado).
8) Não é resiliente → Uma vulnerabilidade em uma aplicação pode comprometer todo o ambiente (efeito dominó).
9) Não é rastreável → Alterações não têm versionamento nem auditoria formal.
10) Não segue boas práticas modernas → Faltam controles de CI/CD, infraestrutura como código (IaC), pipelines de segurança (SonarQube, SAST/DAST), segregação de ambientes e Zero Trust.
11) Dependência excessiva de equipes específicas → O processo depende de um número reduzido de analistas ("gargalos humanos"), gerando risco operacional.
12) Tempo de entrega elevado → Como tudo é manual, desde configuração até testes, o tempo entre desenvolvimento e produção aumenta drasticamente (lead time alto).

B. Avaliação do nível de maturidade em DevSecOps

A maturidade DevSecOps pode ser avaliada por meio de modelos que analisam práticas, processos, ferramentas e aspectos culturais relacionados à integração da segurança no ciclo de desenvolvimento. Entre esses modelos, o DevSecOps Maturity Model (DSOMM), criado pela comunidade OWASP, destaca-se por oferecer um framework estruturado que mede o grau de automação, governança e efetividade das práticas de segurança em ambientes DevOps. Ele permite identificar desde níveis iniciais, com processos manuais, até estágios avançados, figura 5, nos quais a segurança é contínua, automatizada e integrada ao pipeline CI/CD, servindo como guia para evolução e priorização de melhorias [11], [18].

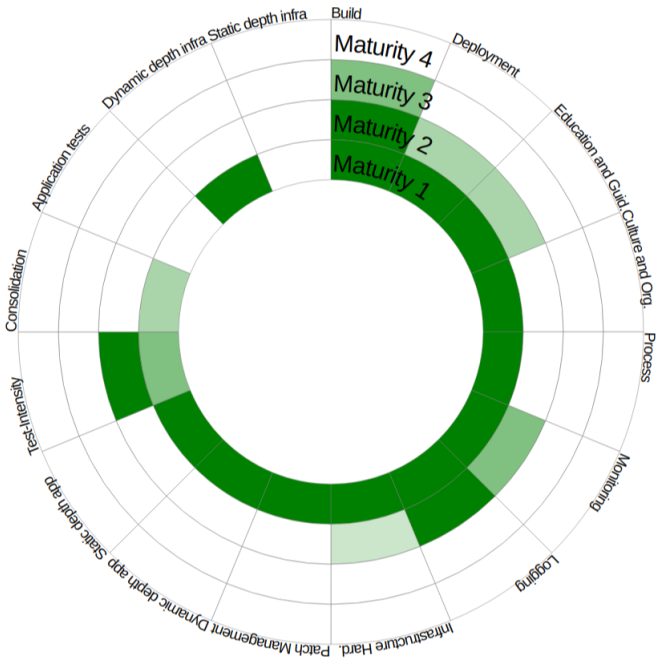

Figura 5. Nível de implementação DSOMM

O OWASP DevSecOps Maturity Model (DSOMM) destaca-se como uma das iniciativas mais completas para incorporar a segurança de forma contínua no ciclo DevOps. O modelo fornece uma estrutura clara e incremental que orienta equipes técnicas na avaliação e melhoria das práticas de segurança, promovendo a integração entre desenvolvimento, operações e segurança [11]. Diferentemente de frameworks tradicionais, o DSOMM alia automação, cultura e métricas, permitindo mensurar a maturidade e priorizar ações com base em evidências [18]. Sua flexibilidade e enfoque prático favorecem a adoção gradual e adaptável a diferentes contextos organizacionais, fortalecendo a cultura de segurança colaborativa e reduzindo barreiras entre áreas técnicas e de segurança [19] .

O DSOMM é dividido em categorias temáticas (domínios) que cobrem todas as áreas relevantes de segurança em DevOps, conforme Tabela III.

Tabela III
CATEGORIAS DSOMM

Categoria	Foco
Construção de pipelines	Segurança integrada ao CI/CD
Gerenciamento de dependências	Verificação de vulnerabilidades em bibliotecas
Gerenciamento de segredos	Gestão segura de senhas, tokens e chaves
Infrastructure as Code	Segurança no provisionamento de infraestrutura via código
IAM (Gerenciamento de identidade e acesso)	Controle de acesso e identidade
Testes de segurança	Testes automatizados de segurança (SAST, DAST, etc.)
Monitoramento e registro de logs	Monitoramento, detecção e resposta a incidentes
Modelagem de ameaças	Modelagem de ameaças no início do desenvolvimento
Segurança de contêineres	Segurança em ambientes com Docker/Kubernetes
Conformidade como Código	Políticas e requisitos de conformidade automatizados

Cada domínio é dividido em níveis de maturidade, geralmente de 1 (inexistente) até 3 ou 4 (altamente maduro) [18], conforme exemplo apresentado na tabela IV:

Tabela IV
NÍVEIS DE MATURIDADE DSOMM

Nível	Descrição
1	Nenhuma prática implementada ou práticas reativas e manuais
2	Práticas iniciais ou pontuais, geralmente manuais
3	Práticas parcialmente automatizadas e com cobertura razoável
4	Práticas maduras, totalmente automatizadas e integradas à cultura da equipe

Para avaliação do nível de maturidade em práticas de segurança integradas ao ciclo de desenvolvimento, foi utilizado o modelo OWASP DevSecOps Maturity Model (DSOMM) como referencial metodológico. A aplicação do modelo foi conduzida por meio de uma abordagem qualitativa baseada em brainstorming estruturado, caracterizado por entrevistas semi-estruturadas com os principais atores envolvidos no ciclo de vida das soluções digitais da instituição.

O *brainstorming* foi realizado de forma colaborativa, com a participação de profissionais das áreas de desenvolvimento de software, infraestrutura de TI e segurança da informação. A proposta central da atividade foi mapear o nível de maturidade atual da organização em cada uma das categorias técnicas previstas no DSOMM, como integração contínua (CI), entrega

contínua (CD), gestão de identidade e acesso (IAM), segurança de containers, entre outras.

Durante as sessões, os participantes contribuíram com percepções baseadas em sua experiência prática, permitindo uma análise mais precisa da situação atual e das lacunas existentes. A escolha por essa abordagem visou promover uma visão integrada e realista do ambiente DevSecOps da instituição, considerando tanto aspectos técnicos quanto culturais e organizacionais. Os resultados foram apresentados abaixo:

Tabela V
NÍVEL DE MATURIDADE IDENTIFICADO NA INSTITUIÇÃO

Categoria	Dev (0-3)	Infra (0-3)	Sec (0-3)	Nível Geral	Observações
Construção de pipelines	2	2	2	2	Pipeline de CI está parcialmente implantado, mas segurança ainda não está integrada ao processo.
Gerenciamento de dependências	2	2	2	2	Gestão de dependências é básica; ausência de escaneamento automatizado ou controle rigoroso.
Gerenciamento de segredos	1	1	1	1	Nenhum controle estruturado identificado para gestão de segredos; representa risco crítico.
Infraestrutura como Código	1	2	2	2	Adoção inicial; boas práticas de segurança ainda são limitadas ou não automatizadas.
Teste de Segurança	1	1	2	1	Testes de segurança ocorrem de forma limitada; parte da responsabilidade está centralizada na equipe de segurança.
IAM	3	3	3	3	Prática mais madura, com controles razoavelmente bem definidos entre as áreas.
Monitoramento e registros de log	3	3	3	3	Monitoramento e registro de eventos estão bem distribuídos e funcionalmente aplicados.
Modelagem de ameaças	1	1	1	1	A atividade é conhecida, mas pouco formalizada ou utilizada de forma contínua.
Segurança de contêiner	1	1	1	1	Não há práticas claras de segurança aplicadas ao uso de containers.
Conformidade como Código	1	1	1	1	Inexistência de automação de conformidade; necessidade de iniciativas nesse domínio.

A avaliação baseada no modelo OWASP DevSecOps Maturity Model (DSOMM) permitiu identificar de forma clara o estágio atual de maturidade da organização em relação à integração de práticas de segurança ao ciclo de desenvolvimento de software. Os resultados demonstram que, embora existam iniciativas positivas nas áreas de gestão de identidade e acesso (IAM) e monitoramento e logging, ainda há deficiências significativas em categorias críticas, como gestão de segredos, segurança de containers e compliance como código, todas com nível de maturidade nulo (1). A análise também evidenciou que a maturidade está distribuída de forma desigual entre as áreas de desenvolvimento, infraestrutura e segurança, o que reforça a necessidade de maior integração entre os times e de uma cultura DevSecOps mais consolidada. A ausência de práticas fundamentais de segurança automatizada compromete não apenas a resiliência dos sistemas, mas também a conformidade com diretrizes do Programa de Privacidade e Segurança da Informação (PPSI) e da legislação vigente, como a LGPD. Portanto, recomenda-se a priorização de ações estratégicas voltadas à automação da segurança, à formalização de processos e ao fortalecimento da colaboração interdisciplinar. A evolução da maturidade DevSecOps será um fator determinante para garantir não apenas a segurança e conformidade das soluções digitais, mas também a agilidade e a confiabilidade dos serviços prestados à população.

C. Validação da arquitetura

Para validar e testar a arquitetura de pipeline CI/CD proposta, foi desenvolvido um laboratório experimental que reproduziu as principais fases de integração, entrega e segurança contínuas, utilizando as tecnologias destacadas no fluxo arquitetural. O objetivo do experimento foi avaliar a aplicabilidade prática da arquitetura, sua eficiência na automação de entregas e a aderência aos princípios de DevSecOps.

O laboratório foi implementado a partir das seguintes tecnologias:

- **GitHub Actions:** utilizado como orquestrador de pipeline, responsável pela execução automatizada das etapas de build, análise de código, testes unitários, empacotamento em contêiner e deploy em ambientes de homologação e produção [3], [12], [13].
- **Docker:** aplicado para o empacotamento das aplicações e suas dependências, garantindo portabilidade, reprodutibilidade e padronização dos ambientes [3], [11], [12].
- **Secrets Management:** configurado no repositório GitHub para o armazenamento seguro de credenciais sensíveis (tokens de acesso, chaves de deploy e credenciais de banco de dados), em conformidade com as boas práticas de segurança [3], [12], [13].
- **SonarQube:** integrado ao pipeline para análise estática de código (SAST), permitindo identificar bugs, vulnerabilidades, code smells e violações de padrões de codificação antes da promoção para ambientes superiores [3], [12], [13].
- **Trivy:** utilizado para análise de vulnerabilidades em imagens Docker e dependências de software (SCA), permitindo detectar componentes inseguros e avaliar a conformidade das imagens de contêiner com políticas de segurança [3], [11], [12].
- **Zaproxy (OWASP ZAP):** empregado na análise dinâmica de segurança (DAST) durante a execução da aplicação, com o objetivo de identificar vulnerabilidades em tempo de execução, como falhas de autenticação, injeções e exposições de dados [3], [11], [12].
- **Kubernetes:** adotado como plataforma de orquestração de contêineres, responsável pela implantação e gerenciamento da aplicação em ambientes de homologação (staging) e produção, oferecendo suporte à escalabilidade, alta disponibilidade e mecanismos automatizados de rollback [3], [11], [12].
- **Prometheus:** implementado para o monitoramento contínuo da infraestrutura e das aplicações, coletando

métricas de desempenho, consumo de recursos e disponibilidade dos serviços implantados [11], [12].

- **Grafana:** utilizado como ferramenta de visualização e análise de métricas, integrado ao Prometheus para a construção de painéis em tempo real, permitindo a identificação proativa de incidentes e tendências operacionais [12], [13].
- **Trend Micro Workload Security:** aplicado como camada de proteção adicional em nível de workload e contêiner, oferecendo detecção de ameaças, varredura de vulnerabilidades, controle de integridade e prevenção de intrusões nos ambientes Kubernetes, reforçando a segurança operacional e a conformidade regulatória [20].

Para abstrair a infraestrutura necessária à execução dos experimentos, foram utilizados serviços de nuvem da DigitalOcean, que proveram os recursos de Kubernetes Managed Cluster e rede de suporte. A arquitetura contemplou também bases de dados PostgreSQL, configuradas separadamente para os ambientes de homologação e produção, a fim de refletir cenários reais de segregação de ambientes. O sistema foi adaptado a partir de um fork (cópia derivada) de um repositório público disponível no GitHub, originalmente desenvolvido por Fabrício Veronez [21], com a finalidade de simular um ambiente de gerenciamento de eventos.

Essa configuração laboratorial possibilitou a simulação de todo o ciclo de vida da aplicação em um cenário controlado, reproduzindo etapas reais de um pipeline DevSecOps. Dessa forma, foi possível observar a efetividade da arquitetura proposta na entrega de software com agilidade, padronização, rastreabilidade e segurança, confirmando sua viabilidade para implantação em contextos institucionais e corporativos.

*1) **Etapa 1 – Início da Integração Contínua (CI) - Fases 1 a 4**:* O pipeline é iniciado a partir de um push ou pull request na branch principal do repositório, conforme figura 6.

Figura 6. Inicio do pipeline CI/CD

Nessa fase, o código-fonte é obtido e submetido a uma análise de qualidade e segurança por meio do SonarQube, responsável pela identificação de vulnerabilidades, más práticas e falhas de segurança no código (SAST), conforme apresentados nas figuras 7 e 8.

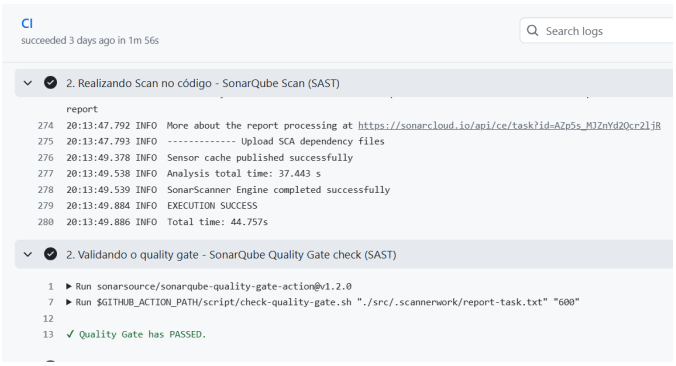

Figura 7. Scan código com SonarQube

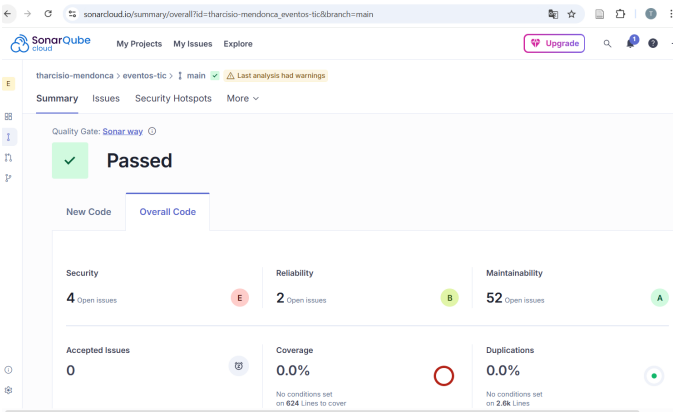

Figura 8. SonarQube validando o código

Após a validação inicial, é gerada a imagem Docker da aplicação e registrada no repositório, figura 9.

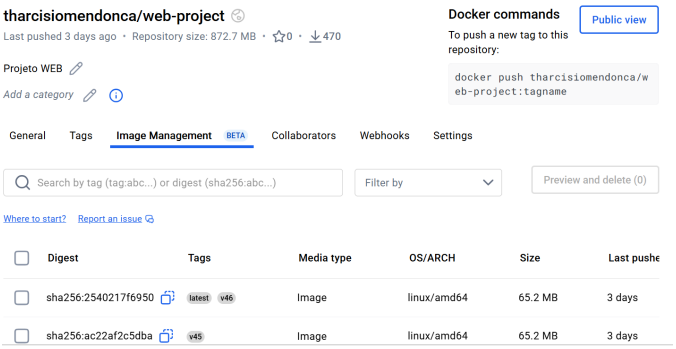

Figura 9. Imagem registrada no repositório

Em seguida, ocorre um novo procedimento de verificação de segurança voltado à análise de componentes de terceiros — como bibliotecas e pacotes de código aberto — utilizando a ferramenta Trivy, que realiza o Software Composition Analysis (SCA) para detectar vulnerabilidades conhecidas em dependências externa, figura 10.

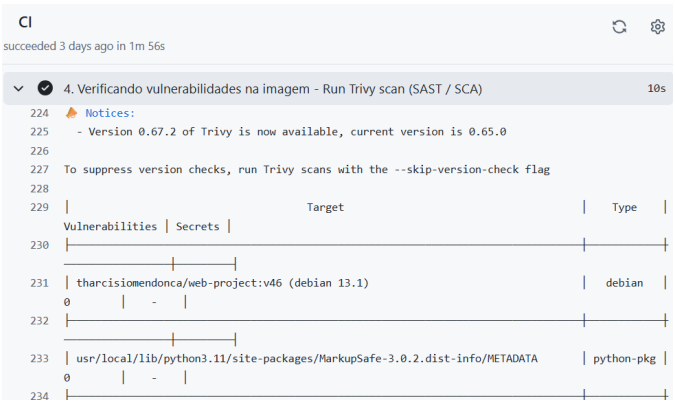

Figura 10. Scan na imagem docker pela ferramenta Trivy

*2) **Etapa 2 - Entrega Contínua (CD) e Testes Dinâmicos - Fase 5 (Homologação):*** Concluída a fase de build, inicia-se o processo de entrega contínua, com o deploy da imagem na infraestrutura de Kubernetes. Após a criação do ambiente de homologação, é executado um scan de vulnerabilidades em tempo de execução utilizando o Zaproxy, ferramenta de Dynamic Application Security Testing (DAST), figura 11. Essa etapa é essencial para identificar vulnerabilidades presentes durante a execução real da aplicação.

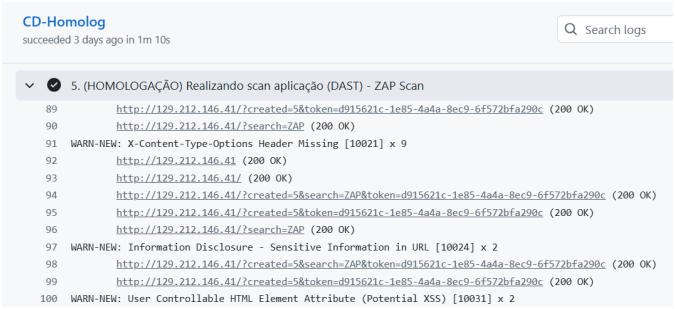

Figura 11. Execução de scan de vulnerabilidades na aplicação em homologação

*3) **Etapa 3 - Aprovação e Publicação em Produção - Fase 5 (Produção):*** Após a validação das análises de segurança (SAST, SCA e DAST), o pipeline é pausado aguardando a aprovação formal de um responsável técnico ou gestor, conforme figura 12.

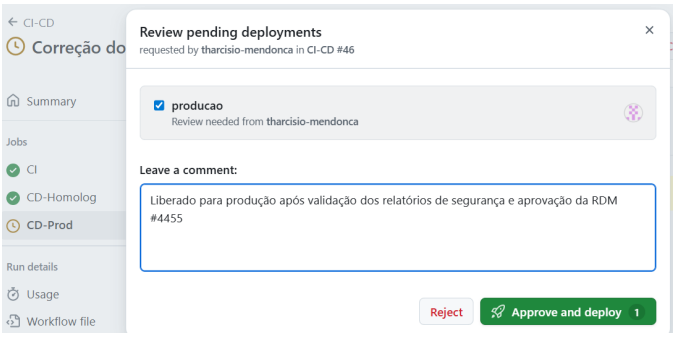

Figura 12. Aguardando autorização para publicação em produção

Mediante aprovação, o sistema é liberado e implantado no ambiente de produção e o pipeline finalizado.

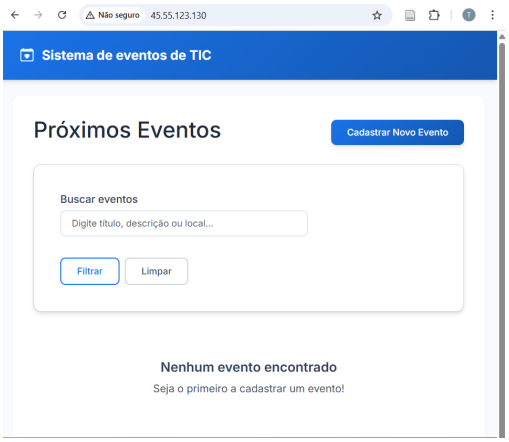

Figura 13. Sistema em produção

*4) **Etapa 4 - Monitoração e Proteção Contínua - Fases 6 e 7:*** O monitoramento de métricas de desempenho e disponibilidade é realizado por meio do Prometheus, figura 14 enquanto os dashboards e painéis analíticos são disponibilizados no Grafana, figura 15.

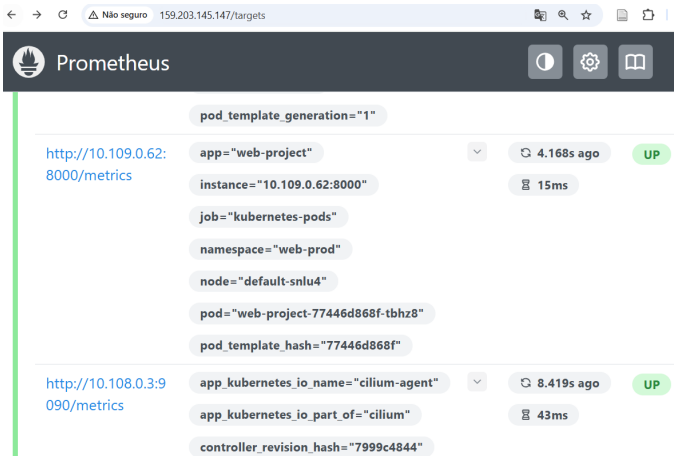

Figura 14. Prometheus - Métricas

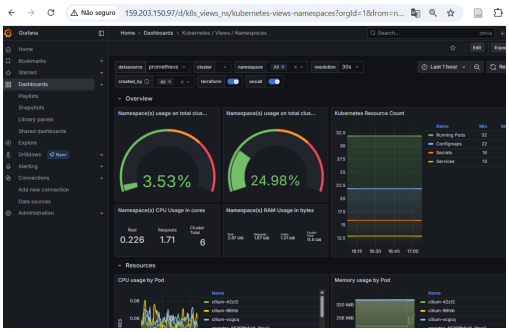

Figura 15. Grafana - Dashboards

A proteção contínua dos ambientes é garantida pela solução Trend Micro Workload Security, que utiliza tecnologia XDR (Extended Detection and Response) para correlacionar eventos e ameaças de segurança em tempo real.

VI. Análise dos Resultados

O modelo de pipeline apresentado pela arquitetura proposta incorpora práticas de Integração Contínua (CI) e Entrega Contínua (CD) utilizando Docker, Kubernetes, SonarQube, Trivy, Zaproxy, GitHub Actions e ferramentas de monitoramento. Esse arranjo técnico está diretamente associado ao paradigma DevSecOps, em que segurança, qualidade e agilidade são tratadas de forma integrada.

Os resultados apresentados podem ser organizados em cinco dimensões, com mapeamento aos controles do CIS v8 [22] e às diretrizes do PPSI (Programa de Privacidade e Segurança da Informação).

A. Padronização e portabilidade

O uso de Docker garante que os artefatos sejam encapsulados em imagens imutáveis e portáveis. Essa prática elimina discrepâncias entre ambientes de desenvolvimento, homologação e produção.

- **Alinhamento ao CIS Controls v8:** Controle 04 - Configuração segura de ativos corporativos e software
- **Alinhamento ao PPSI:** Redução de riscos operacionais associados à falta de padronização de ambientes.

Resultado: confiabilidade e previsibilidade nos processos de entrega, com diminuição de falhas decorrentes de incompatibilidades de infraestrutura.

1) Qualidade e segurança do código (SonarQube): O SonarQube realiza análise estática do código fonte, identificando vulnerabilidades, bugs, violações de padrões e code smells antes da promoção do build. Essa validação atua de forma complementar aos testes unitários, reforçando a adoção de práticas de Secure Coding e "Shift-Left Security".

- **Alinhamento ao CIS Controls v8:** Controle 16 – Segurança de aplicações.
- **Alinhamento ao PPSI:** Diretriz de desenvolvimento seguro e prevenção contra falhas de software exploráveis.

Resultado: mitigação de riscos cibernéticos desde a fase de desenvolvimento, prevenindo a introdução de vulnerabilidades em produção.

2) Eficiência operacional e agilidade: A automação do pipeline por meio do GitHub Actions reduz o tempo de entrega (time-to-market), permitindo que cada modificação de código seja automaticamente validada, construída, testada e preparada para deploy. A orquestração em Kubernetes facilita escalabilidade e resiliência.

- **Alinhamento ao CIS Controls v8:** Controle 8 – Gestão de registros de auditoria.
- **Alinhamento ao PPSI:** Garantia de rastreabilidade dos processos de implantação.

Resultado: aumento da velocidade de entregas sem comprometer a qualidade ou a segurança do software.

3) Governança, confiabilidade e resposta a incidentes: A arquitetura prevê aprovação manual em etapas críticas, alinhando-se a boas práticas de auditoria e gestão de mudanças. Adicionalmente, a presença de mecanismos de rollback garante continuidade de serviços mesmo em caso de falhas em produção.

- **Alinhamento ao CIS Controls v8:** Controle 17 – Gestão de respostas a incidentes.
- **Alinhamento ao PPSI:** Atendimento às diretrizes de governança, responsabilidade e controle no ciclo de vida da informação.

Resultado: maior confiabilidade do processo de deploy, suporte à conformidade regulatória e capacidade de resposta rápida a falhas.

4) Observabilidade, monitoramento e melhoria contínua: A integração com ferramentas de segurança (XDR, SIEM) e observabilidade (APM, métricas e logs) garante visibilidade em tempo real sobre desempenho e ameaças. Essa camada de observabilidade alimenta o ciclo de melhoria contínua, permitindo ajustes proativos e otimização da arquitetura.

- **Alinhamento ao CIS Controls v8:** Controle 6 – Gestão de controle de acesso e Controle 8 – Gestão de registros de auditoria.
- **Alinhamento ao PPSI:** Monitoramento contínuo de riscos e avaliação periódica de controles de segurança.

Resultado: maior resiliência organizacional, capacidade de detecção precoce de incidentes e fortalecimento da postura de segurança.

VII. Conclusão

A arquitetura proposta de DevSecOps em nuvem, validado em laboratório com GitHub Actions, Docker, Kubernetes, SonarQube, Trivy, Zaproxy entre outras, demonstrou sua viabilidade ao integrar segurança, automação e governança em todo o ciclo de vida do software. Os resultados evidenciaram ganhos em padronização, portabilidade, qualidade do código, eficiência operacional e conformidade com o PPSI e CIS Controls v8.

A adoção de práticas de DevSecOps mostrou-se essencial para reduzir vulnerabilidades, aumentar a confiabilidade das entregas e fortalecer a postura institucional frente à segurança da informação e à privacidade de dados. Recomenda-se, como continuidade, a ampliação da arquitetura com Infraestrutura como Código e políticas de Zero Trust, visando maior maturidade e resiliência digital.

Entretanto, é oportuno destacar que a implantação de uma arquitetura DevSecOps demanda um grande esforço de mudança cultural e capacitação contínua de toda a equipe envolvida.

Como trabalhos futuros, recomenda-se o aprimoramento da arquitetura proposta por meio da integração do pipeline a soluções baseadas em Inteligência Artificial (IA), de modo a ampliar a automação, a detecção proativa de vulnerabilidades e a eficiência operacional.

REFERÊNCIAS

[1] A. F. de Notícias, "Fiocruz faz 120 anos diante do maior desafio do século 21 | Portal Fiocruz," May 2020.

[2] Oluwatosin Oluwatimileyin Abiona, Oluwatayo Jacob Oladapo, Oluwole Temidayo Modupe, Oyekunle Claudius Oyeniran, Adebunmi Okechukwu Adewusi, and Abiola Moshood Komolafe, "The emergence and importance of DevSecOps: Integrating and reviewing security practices within the DevOps pipeline," *World Journal of Advanced Engineering Technology and Sciences*, vol. 11, pp. 127–133, Mar. 2024.

[3] M. M. Pinto, "AppSeg: Um Sistema para Apoio à Adoção de DevSecOps,"

[4] F. M. Constante, R. Soares, M. Pinto-Albuquerque, D. Méndez, and K. Beckers, "Integration of Security Standards in DevOps Pipelines: An Industry Case Study," in *International Conference on Product-Focused Software Process Improvement*, vol. 12562, pp. 434–452, PROFES, 2020. arXiv:2105.13024 [cs].

[5] R. C. Thota, "Cloud-Native DevSecOps: Integrating Security Automation into CI/CD Pipelines," *IJIRCT*, Dec. 2024. Publisher: Zenodo.

[6] A. S. A. Alghawli and T. Radivilova, "Resilient cloud cluster with DevSecOps security model, automates a data analysis, vulnerability search and risk calculation," *Alexandria Engineering Journal*, vol. 107, pp. 136–149, Nov. 2024.

[7] S. Rangaraju, D. S. Ness, and R. Dharmalingam, "Incorporating AI-Driven Strategies in DevSecOps for Robust Cloud Security," *International Journal of Innovative Science and Research Technology*, Dec. 2023. Publisher: Zenodo.

[8] A. Verdet, M. Hamdaqa, L. D. Silva, and F. Khomh, "Exploring Security Practices in Infrastructure as Code: An Empirical Study," *Empirical Software Engineering*, vol. 30, p. 74, May 2025. arXiv:2308.03952 [cs].

[9] R. S. Wazlawick, *Metodologia de pesquisa para ciência da computação*. GEN LTC, 2014. OCLC: 902734376.

[10] T. Hsu, *Hands-on security in DevOps: ensure continuous security, deployment, and delivery withDevSecOps*. Erscheinungsort nicht ermittelbar: Packt Publishing, 1st ed ed., 2018.

[11] OWASP, "OWASP Devsecops Maturity Model | OWASP Foundation."

[12] L. Pessol, "Como começar com DevSecOps," July 2024.

[13] "OWASP DevSecOps Guideline - v-0.2 | OWASP Foundation."

[14] D. S. A. d. Freitas, A. A. d. Oliveira, E. D. Moreno, and G. J. F. d. Silva, "DevSecOps Practices for GDPR, HIPAA or LGPD Compliance in Software Development: A Systematic Review," in *Simpósio Brasileiro de Sistemas de Informação (SBSI)*, pp. 145–153, SBC, May 2025. ISSN: 0000-0000.

[15] A. Ibrahim, A. H. Yousef, and W. Medhat, "DevSecOps: A Security Model for Infrastructure as Code Over the Cloud," in *2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*, pp. 284–288, May 2022.

[16] S. d. G. D. SGD, "Guia de Gerenciamento de Vulnerabilidades," 2022.

[17] J. Alonso, R. Piliszek, and M. Cankar, "Embracing IaC Through the DevSecOps Philosophy: Concepts, Challenges, and a Reference Framework," *IEEE Software*, vol. 40, pp. 56–62, Jan. 2023.

[18] R. Brasoveanu, Y. Karabulut, and I. Pashchenko, "Security Maturity Self-Assessment Framework for Software Development Lifecycle," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, ARES '22, (New York, NY, USA), pp. 1–8, Association for Computing Machinery, 2022.

[19] A. Krasnov and D. R. Maiti, "Overview of DevSecOps frameworks for Software Development Lifecycle and its current limitations," 2024.

[20] "Solução Cloud Workload Security - Trend Vision One™ | Trend Micro (BR)."

[21] F. Veronez, "Encontros Tech."

[22] C. F. I. S. CIS, "Controles CIS Versão 8," 2021.