UNIVERSIDADE ESTADUAL DO PIAUÍ
CENTRO DE TECNOLOGIA E URBANISMO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Robério Guilherme Sousa da Silva

Implantação Pioneira de Integração Contínua e Entrega Contínua (CI/CD) na FMS: Desenvolvimento de uma Esteira Automatizada com Deploy via SSH em Servidor VPS

TERESINA
2025

Robério Guilherme Sousa da Silva

Implantação Pioneira de Integração Contínua e Entrega Contínua (CI/CD) na FMS: Desenvolvimento de uma Esteira Automatizada com Deploy via SSH em Servidor VPS

Monografia de Trabalho de Conclusão de Curso apresentado na Universidade Estadual do Piauí – UESPI como parte dos requisitos para conclusão do Curso de Bacharelado em Ciência da Computação.

Orientador: Prof. Dr. Carlos Giovanni Nunes de Carvalho

TERESINA

2025

S586i Silva, Roberio Guilherme Sousa da.
Implantação pioneira de Integração Contínua e Entrega Contínua (CI/CD) na FMS: desenvolvimento de uma esteira automatizada com Deploy via SSH em servidor VPS / Roberio Guilherme Sousa da Silva. - 2025.
30 f.: il.

Monografia (graduação) - Universidade Estadual do Piauí-UESPI, Bacharelado em Ciência da Computação, Campus Poeta Torquato Neto, Teresina-PI, 2025.
"Orientador: Prof. Dr. Carlos Giovanni Nunes de Carvalho".

1. Entrega contínua. 2. Integração contínua. 3. Deploy Automatizado. 4. VPS. 5. SSH. I. Carvalho, Carlos Giovanni Nunes de . II. Título.

CDD 004.07

Ficha elaborada pelo Serviço de Catalogação da Biblioteca da UESPI
Nayla Kedma de Carvalho Santos (Bibliotecário) CRB-3ª/1188

Implantação Pioneira de Integração Contínua e Entrega Contínua (CI/CD) na FMS: Desenvolvimento de uma Esteira Automatizada com Deploy via SSH em Servidor VPS

Robério Guilherme Sousa da Silva

Monografia de Trabalho de Conclusão de Curso apresentado na Universidade Estadual do Piauí – UESPI como parte dos requisitos para conclusão do Curso de Bacharelado em Ciência da Computação.

gov.br
Documento assinado digitalmente
CARLOS GIOVANNI NUNES DE CARVALHO
Data: 09/07/2025 10:32:06-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Carlos Giovanni Nunes de Carvalho, Dsc.
Orientador

Nota da Banca Examinadora: 8,5

Banca Examinadora:

gov.br
Documento assinado digitalmente
CARLOS GIOVANNI NUNES DE CARVALHO
Data: 09/07/2025 09:37:30-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Carlos Giovanni Nunes de Carvalho, Dsc.
Presidente

gov.br
Documento assinado digitalmente
THIAGO CARVALHO DE SOUSA
Data: 09/07/2025 15:05:40-0300
Verifique em https://validar.iti.gov.br

Thiago Carvalho de Sousa, Dr.
Membro

gov.br
Documento assinado digitalmente
SERGIO BARROS DE SOUSA
Data: 09/07/2025 19:38:11-0300
Verifique em https://validar.iti.gov.br

Sérgio Barros de Sousa, Dr.
Membro

AGRADECIMENTOS

Gostaria de expressar minha sincera gratidão à minha família, que sempre esteve ao meu lado, oferecendo apoio, incentivo e compreensão em todos os momentos desta jornada acadêmica.

Agradeço também à equipe da FMS, que contribuiu de maneira significativa para a construção deste trabalho, compartilhando conhecimento, oferecendo orientação e auxiliando de forma dedicada na superação dos desafios encontrados ao longo do desenvolvimento deste estudo.

Registro, ainda, meu reconhecimento e agradecimento à equipe de docentes da Universidade Estadual do Piauí (UESPI), pela excelência no ensino, pela dedicação e pelo compromisso em formar profissionais capacitados, que contribuíram diretamente para a minha formação acadêmica e pessoal.

Estendo meus agradecimentos a todos os amigos de turma, com quem compartilhei essa caminhada acadêmica, marcada por momentos de aprendizado, companheirismo e união, que tornaram essa experiência ainda mais significativa.

A todos vocês, meu muito obrigado!

“Do. Or do not. There is no try“
(Yoda, Star Wars: Episode V – The Empire Strikes Back, 1980)

.

RESUMO

Atualmente, as demandas por atualizações rápidas e entregas contínuas de *software* impulsionam o uso de metodologias e ferramentas que automatizem processos, aumentem a segurança e otimizem o fluxo de desenvolvimento. Contudo, a Fundação Municipal de Saúde enfrenta problemas no processo de *deploy* de seus serviços, o que compromete a agilidade e a confiabilidade das atualizações. Nesse contexto, este trabalho apresenta um estudo de caso sobre a implementação de uma esteira de integração e entrega contínua em ambiente Virtual Private Server utilizando conexões seguras via SSH. Para isso, foram utilizadas ferramentas como Bitbucket Pipelines, bibliotecas de linter, testes unitários e arquivos shell script responsáveis pelo gerenciamento das operações de atualização e publicação. A metodologia adotada possui caráter exploratório, descrevendo a construção da esteira, a integração das ferramentas e o processo de *deploy* remoto automatizado.

Palavras-chaves: Entrega Contínua. Integração Contínua. Deploy Automatizado. VPS. SSH.

ABSTRACT

Currently, the demand for rapid updates and continuous software deliveries drives the use of methodologies and tools that automate processes, enhance security, and optimize the development workflow. However, the Municipal Health Foundation faces issues in the deployment process of its services, which compromises the agility and reliability of updates. In this context, this work presents a case study on the implementation of a continuous integration and delivery pipeline in a Virtual Private Server environment using secure SSH connections. For this purpose, tools such as Bitbucket Pipelines, linter libraries, unit tests, and shell script files responsible for managing update and publishing operations were employed. The adopted methodology is exploratory in nature, describing the construction of the pipeline, the integration of tools, and the automated remote deployment process.

Keywords: Continuous Integration. Continuous Delivery. Automated Software Deployment. VPS. SSH.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CI	*Continuous Integration*
CD	*Continuous Delivery*
VPS	*Virtual Private Server*
SSH	*Secure Shell*
Brasscom	*Associação das Empresas de Tecnologia da Informação e Comunicação*
ABES	*Associação Brasileira das Empresas de Software*
TI	*Tecnologia da Informação*
FMS	*Fundação Municipal de Saúde*
IP	*internet protocol*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, o avanço das tecnologias de desenvolvimento de *software* e a adoção crescente de metodologias ágeis transformaram significativamente a forma como aplicações são concebidas. No desenvolvimento tradicional, equipes distintas desempenham tarefas específicas de acordo com os seus níveis de responsabilidade, essa separação torna o processo mais demorado e dificulta a comunicação entre os times (Santos, 2023). Em meio a esse cenário, práticas como *Continuous Integration* (CI) e *Continuous Delivery* (CD) surgiram como soluções eficazes para reduzir o tempo entre o desenvolvimento e a disponibilização de novas versões de aplicações.

Embora ferramentas de automação e boas práticas estejam cada vez mais presentes em grandes organizações, muitos desenvolvedores e pequenas empresas que utilizam *Virtual Private Server* (VPS), uma vez que esse tipo de hospedagem oferece grande versatilidade e controle aliado a um baixo custo (Gea; Lase; Syamsudin, 2023), acabam enfrentando dificuldades para implementar soluções de *deploy* contínuo eficientes e seguras.

Esse cenário se intensifica com a necessidade de garantir conexões protegidas e fluxos automatizados adaptados às particularidades de cada ambiente. A utilização do *Secure Shell* (SSH) para autenticação e execução de comandos remotos, embora consolidada, exige configurações criteriosas para preservar a segurança e a integridade do ambiente, pois o protocolo é estruturado em três camadas de proteção, proporcionando *login* remoto e serviços de rede seguros sobre redes inseguras (Ylonen; Lonvick, 2006). Nesse contexto, a integração de *pipelines* de CI/CD com VPS via SSH surge como uma solução promissora.

Portanto, durante a análise inicial, foi identificado um problema relacionado à ausência de uma esteira de CI/CD nos projetos da Fundação Municipal de Saúde (FMS), o que compromete a agilidade e a segurança nos processos de atualização de sistemas. Diante disso, propõe-se estruturar, implementar e validar uma solução acessível, segura e replicável, contribuindo para a otimização dos processos de *deploy* em ambientes controlados.

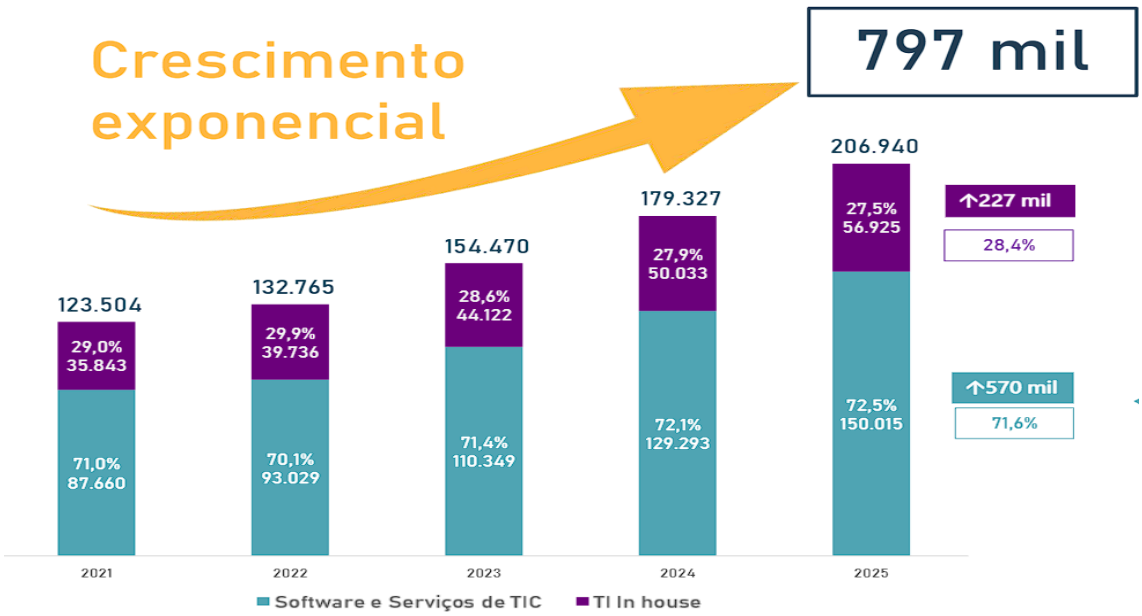
1.1 Justificativa

A revolução tecnológica tem se tornado cada vez mais evidente, com uma crescente demanda por *softwares* que atendam a necessidades específicas, o crescimento do mercado de trabalho e o desenvolvimento de tecnologias.

O setor de Tecnologia da Informação (TI) vive um momento de expansão acelerada no Brasil, consolidando-se como um dos principais motores de transformação digital e competitividade empresarial. De acordo com o Estudo Mercado Brasileiro de *Software* – Panorama e Tendências 2025, apresentado pela ABES (Associação Brasileira das Empresas de Software), o mercado nacional de TI registrou um crescimento expressivo de 13,9% em 2024, ultrapassando a média global de 10,8%, com previsão de alcançar 9,5% de crescimento em 2025, acima da expectativa mundial de 8,9%. Esse desempenho reafirma o Brasil como o maior na área TI na América Latina e o coloca na 10° posição no *ranking* mundial de investimentos em tecnologia, com forte destaque para áreas como inteligência artificial, segurança cibernética e infraestrutura digital.

Paralelamente a esse cenário, o mercado brasileiro de tecnologia apresentou recuperação e crescimento sólido após a instabilidade vivida em 2023. Segundo levantamento da *Advance Consulting*, o setor obteve 21% de crescimento em 2024, impulsionado pela demanda por soluções em Inteligência Artificial, nuvem computacional, segurança da informação e automação de processos. Além disso, a Brasscom (Associação das Empresas de Tecnologia da Informação e Comunicação) estima a criação de cerca de 800 mil vagas de trabalho até 2025 como pode ser observado na figura 1, enquanto a carência de profissionais qualificados permanece um desafio, o que reforça a importância de soluções automatizadas que aumentem a produtividade e otimizem a infraestrutura tecnológica das empresas.

Figura 1 – Demanda de novos talentos em tecnologia em 5 anos



Fonte: Demanda de Talentos em TIC e Estratégia Σ TCEM - Brasscom

Neste contexto, a modernização da FMS, alinhando-a aos novos padrões de

mercado, torna-se imprescindível. A automação de processos de *deploy* por meio de *pipelines* de Integração Contínua e Entrega Contínua (CI/CD), aliada a conexões seguras, configura-se como uma estratégia essencial para organizações que buscam acompanhar a rápida evolução das atualizações de *software*, garantindo segurança, estabilidade e eficiência em múltiplos ambientes.

1.2 Objetivo

Esta seção apresenta o objetivo geral do trabalho e os desdobra em objetivos específicos necessários para sua realização.

1.2.1 Objetivo Geral

Implementar uma esteira de CI/CD para automatizar o processo de *deploy* em um servidor VPS, utilizando conexão segura via SSH na FMS.

1.2.2 Objetivos Específicos

- Configurar conexão SSH entre VPS e repositório.
- Configurar o *pipeline* CI/CD.
- Adaptar o *pipeline* para análise estática do código e execução de testes unitários.
- Viabilizar o *deploy*.

2 REFERENCIAL TEÓRICO

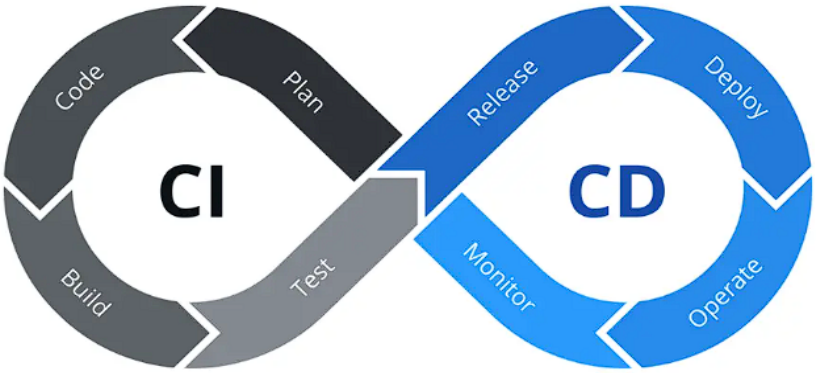
Para a execução deste projeto, alguns conceitos são de suma importância, pois os mesmos funcionam como alicerce para a compreensão das tecnologias e referências que foram utilizadas.

2.1 Ciclo CI/CD

O ciclo de *Continuous Integration* e *Continuous Delivery* é uma prática de desenvolvimento de *software* que se refere à Integração Contínua e Entrega Contínua. Essas práticas são parte essencial de metodologias ágeis e têm como objetivo automatizar o processo de desenvolvimento através de ferramentas, dessa forma otimizando testes e implantação de *software*(Sales, 2025).

Portanto, CI/CD é uma maneira eficaz de melhorar a qualidade, a eficiência e a colaboração no desenvolvimento de *software*, com uma abordagem moderna para o desenvolvimento de *software*, sendo essencial para equipes que buscam otimizar seus processos de entrega de *software*(Habbema, 2023a). Geralmente esse ciclo de trabalho é atrelado ao símbolo do infinito para representar a ideia de continuidade como visto na figura 2.

Figura 2 – Ciclo CI/CD



Fonte: O que é CI/CD? Objetivos e funcionalidades(2025) - sydle

2.1.1 Integração Contínua

A Integração Contínua ou *Continuous Integration* é um processo importante no desenvolvimento de *software* que oferece aos desenvolvedores a vantagem de criar versões consistentes do projeto e identificar possíveis problemas o mais cedo e rápido

possível, a CI atinge suas metas com a automatização do processo de integração do código, proporcionando um ciclo de desenvolvimento consistente, com interações mínimas do usuário (Virtanen, 2021).

Aplicar uma CI eficiente requer o comprometimento da equipe, já que é essencialmente um conjunto de diretrizes para a produção de um projeto. Para isso, a equipe deve trabalhar de forma coesa ao longo do desenvolvimento do *software*, uma vez que os servidores de CI armazenam o código-fonte em um único repositório ou uma *branch* dentro desse repositório. A versão do código-fonte obtida é usada para criar um *build* de integração. Essa *build* deve ser revisada frequentemente para detectar erros o mais cedo possível (Duvall; Matyas; Glover, 2007).

Uma CI funcional permite que os desenvolvedores tenham uma maior visibilidade do projeto e mais confiança em seu produto, fornecendo informações sobre o cenário presente do projeto com base na qualidade geral dos versionamentos. A adoção de CI proporciona diversas vantagens, como a redução dos ciclos de *release*, a melhoria da qualidade do software, o aumento da produtividade das equipes de desenvolvimento e a obtenção de *feedback* rápido e contínuo sobre o estado das alterações realizadas. Tais benefícios decorrem, sobretudo, da automação dos processos de *build* e testes, possibilitando a detecção precoce de defeitos e o alinhamento constante entre as entregas parciais e os requisitos do produto final(Shahin; Babar; Zhu, 2017).

2.1.2 Entrega Contínua

Entrega Contínua ou *Continuous Delivery* é um processo de desenvolvimento de *software* que amplia a Integração Contínua ao automatizar a liberação do *software* em ambiente específico. Nessa etapa, cujo objetivo principal é manter o software em um estado sempre implantável, ou seja, o sistema está continuamente validado, testado e pronto para ser colocado em produção a qualquer momento(Shahin et al., 2017).

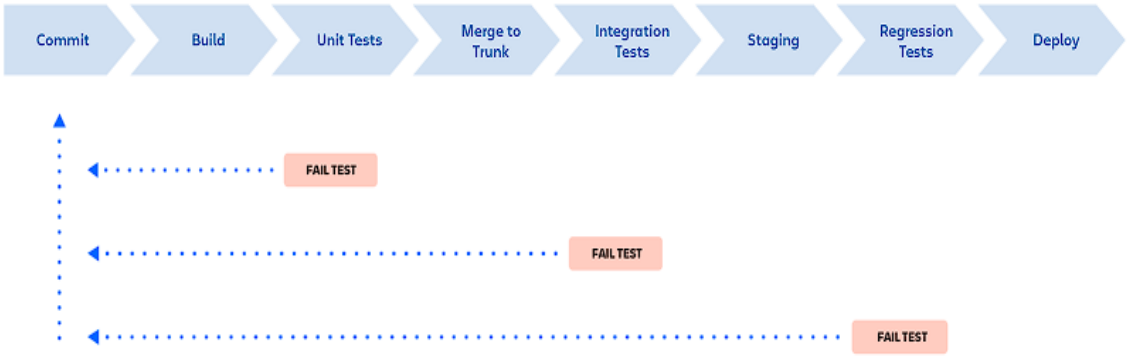
Os benefícios de utilizar essa prática estão relacionados à capacidade de levar qualquer tipo de mudança como novas funcionalidades, alterações de configuração, correções de *bugs* e experimentos, de forma segura, rápida e sustentável para produção ou para as mãos dos usuários(Arachchi; Perera, 2018).

2.2 Pipeline

Um *pipeline* é um conjunto de processos e ferramentas automatizados que permitem que os desenvolvedores trabalhem de forma coesa para criar e implantar código em um ambiente de produção. Ele geralmente inclui automação de compilação/integração contínua, testes automatizados, validação e geração de relatórios. Tais processos são apresentados na Figura 3, podendo incluir uma ou mais etapas manu-

ais que exigem intervenção humana antes que o código possa avançar(Hall, 2021). No caso de uma falha, o *pipeline* é interrompido e um *feedback* é fornecido ao desenvolvedor.

Figura 3 – Fases pipeline



Fonte: Atlassian DevOps Pipeline (2020) - Atlassian

O artigo de Hall(Hall, 2021) afirma que cada organização possui tecnologias diferentes que podem impactar no processo, desta forma tornando cada *pipeline* único. Porém, a maioria das organizações utiliza componentes fundamentais semelhantes. Cada etapa é avaliada quanto ao sucesso antes de seguir para a próxima etapa.

2.3 Protocolo SSH

O Protocolo SSH ou *Secure Shell*, foi criado para resolver problemas de segurança, criando uma conexão e comunicação entre o cliente e o servidor de forma segura, por meio da criptografia das informações, impedindo que dados interceptados possam ser lidos, uma vez que estão cifrados(Gaspar, 2025).

Além de utilizar métodos de autenticação por chave, senha ou ambos, assegurando que apenas usuários autorizados tenham acesso ao sistema remoto, acrescenta-se que o SSH permite a criação de túneis seguros para a transferência de outros protocolos, mantendo a integridade e confidencialidade das conexões(Habbema, 2023b).

2.4 Servidor Privado Virtual

Um Servidor Privado Virtual ou *Virtual Private Server*, funciona como um ambiente virtual isolado em um servidor físico extremamente potente, esse é dividido, criando vários servidores isolados virtualmente. Embora compartilhem recursos físicos, mecanismos como, trafego de rede, memória RAM, espaço em disco e processamento são totalmente interindependentes(Lima, 2023).

3 METODOLOGIA

No presente capítulo, descreve-se a metodologia aplicada na condução deste trabalho acadêmico. O enfoque metodológico recai sobre a implementação de uma esteira automatizada, a orquestração do ambiente de *deploy*, a estruturação do pipeline e o estabelecimento da comunicação via SSH entre o repositório e o servidor VPS.

3.1 Orquestração do Ambiente

O ambiente de desenvolvimento e execução para este trabalho, consistiu no provisionamento de uma VPS configurada com sistema operacional Linux, 4 CPUs, 2 GB de memória RAM e 128 GB de espaço em disco. Para o armazenamento do código fonte foi escolhido o *Bitbucket* como repositório de código, foi selecionado o plano *Standard*, com custo de US$ 3,30 por usuário. Utilizou-se a um um projeto *web* escrito na linguagem *Ruby*, utilizando a *framework Ruby on Rails*.

A definição das fases que compõem o *pipeline* se deu em 3 *steps* principais. o primeiro *step* de *analyze* que realiza a análise estática do código-fonte utilizando a biblioteca *rubocop*. O segundo *step* de *tests* onde ocorre a execução dos testes automatizados através da biblioteca *rubocop-rspec*, para assegurar a cobertura total do projeto foram desenvolvidos testes do tipo *request*, *policy*, *routes*, *system* e *view*. Por último, procede-se ao *step* de *deploy*, onde ocorre o processo de implantação do projeto através da biblioteca *mina*.

3.2 Conexão SSH do Repositório com a VPS

Para garantir a comunicação segura entre a VPS e o repositório de código hospedado no *Bitbucket*, é necessária a realização de uma troca de chaves públicas entre as duas partes. Esse procedimento consiste na geração de um par de chaves na VPS, onde a chave pública deve ser adicionada ao repositório no *Bitbucket*, possibilitando que o servidor remoto autentique suas conexões sem a necessidade de senhas. Da mesma forma, a chave pública disponibilizada pelo *Bitbucket* pode ser armazenada na VPS.

O sistema operacional Linux possuí o pacote SSH instalado. O primeiro passo é criar um par de chaves na VPS através do comando `ssh-keygen`, o par de chaves ficou no sub-diretório `.ssh/`, para exibir o conteúdo da chave pública gerada, utiliza-se o comando `cat` seguido do caminho do arquivo, como em: `cat ~/.ssh/id_rsa.pub`,

possibilitando a visualização e a cópia da chave.

Para gerar um par de chaves SSH diretamente pela interface do Bitbucket, basta acessar as Configurações do Repositório, localizar a seção *Pipelines* e selecionar Chaves SSH. Nessa área, o botão Gerar chaves cria automaticamente um novo par de chaves para autenticação. Na mesma página, na seção *Known Hosts*, o endereço *internet protocol* (IP) da VPS junto da chave pública da VPS deve ser adicionada no campo *host address* e, em seguida, deve-se clicar em *Fetch* para registrar a VPS como um *host* confiável.

Após obter a chave pública do repositório, é necessário acessar a VPS e, dentro do diretório `.ssh`, incluir essa chave no arquivo `authorized_keys`. A fim de garantir a segurança da comunicação via SSH utiliza-se os comandos `chmod 700 ~/.ssh` e `chmod 600 ~/.ssh/authorized_keys`, que definem as permissões adequadas para leitura, gravação e execução, evitando acessos não autorizados. Dessa forma, será possível estabelecer uma conexão SSH segura e automatizada entre o repositório e a VPS.

3.2.1 Desenvolvimento do CI

A primeira etapa do CI foi adicionada a biblioteca *rubocop*, que realiza a análise estática do código-fonte. Durante a execução do *pipeline*, as dependências necessárias do projeto são instaladas e ao final desse processo, a biblioteca de *Linter* é invocada para realizar uma última verificação do código como pode ser visto na Tabela 1, garantindo que ele esteja adequado aos padrões.

Tabela 1 – Configuração do Step de analyze

Configuração do Linter no Pipeline
steps: - step: &analyze name: Run Linter caches: - bundler script: - apt update -y && apt install -y build-essential - apt install -y build-essential libpq-dev postgresql-client xvfb wget - bundle config set path ’vendor/bundle’ - bundle install - bundle exec rubocop app/

Fonte: Elaborado pelo autor (2025).

Na segunda etapa do processo de CI, são construídos e executados os tes-

tes unitários do projeto. Para isso, foi incorporada a biblioteca *rubocop-rspec*. Foram desenvolvidos testes do tipo *request*, *policy*, *routes*, *system* e *view*, assegurando a cobertura total do projeto.

Durante a execução do *pipeline*, o ambiente da aplicação é montado virtualmente, com a instalação de dependências e configuração de testes. Ao final, a biblioteca *rubocop-rspec* é executada conforme ilustrado na Tabela 2.

Tabela 2 – Configuração do Step de Tests

Configuração de Testes no Pipeline
- step: &tests name: Run RSpec caches: - bundler services: - postgres - redis script: - curl -sL https://deb.nodesource.com/setup_16.x | bash - - apt-get update -yq - apt-get -o dir::cache::archives=”APT_CACHE_DIR”install -y apt-transport-https build-essential cmake nodejs software-properties-common unzip - wget -q -O - https://dl.yarnpkg.com/debian/pubkey.gpg | apt-key add - - echo ”deb https://dl.yarnpkg.com/debian/ stable main» /etc/apt/sources.list.d/yarn.list - apt-get update -yq - apt-get -o dir::cache::archives=”APT_CACHE_DIR”install -y yarn - yarn install - export RAILS_ENV - export VACINAS_DATABASE_HOST - export DATABASE_URL - export REDIS_URL - export EMAIL_ADDRESS_RECOVERY - gem install bundler -v 2.4.20 - bundle config set path ’vendor/bundle’ - bundle update –bundler - bundle install - bin/rails db:create - bin/rails db:migrate - bin/rails db:seed - bundle exec rspec spec/

Fonte: Elaborado pelo autor (2025).

3.2.2 Desenvolvimento do CD

Primeiramente é desenvolvido o arquivo que ficara dentro do projeto, que consiste basicamente em acessar, via comandos de terminal, o diretório onde se encontra o segundo *script* na VPS, Como mostrado na tabela 3, executando-o em sequência.

Tabela 3 – Script de Execução

Configuração do Script de Execução
steps: - step: &deploy name: Run Deploy script: - #!/bin/bash - echo ”Deploy script started” - cd /home/fmsti - sh pull.sh - echo ”Deploy script finished execution”

Fonte: Elaborado pelo autor (2025).

Na etapa seguinte, o *pipeline* realiza uma chamada de execução responsável por transferir e executar os comandos do primeiro *shell script* na VPS, como evidenciado na tabela 4.

Tabela 4 – Configuração do Step de Deploy

Configuração do Deploy no Pipeline
steps: - step: &deploy name: deploy size: 4x runtime: cloud: atlassian-ip-ranges: true arch: arm script: - cat deploy_vps.sh | ssh $SSH_USER@$SERVER ”bash -s” - echo ”Deploy step finished”

Fonte: Elaborado pelo autor (2025).

Por fim, o segundo *script* acessa a pasta do projeto na VPS, atualiza os arquivos com a *branch* de *deploy* e executa o comando responsável pelo *deploy* da aplicação, pode-se verificar na tabela 5.

Tabela 5 – Script VPS

Configuração do Deploy na VPS
script: - #!/bin/bash -ex - cd ./nome-do-sistem - git pull origin TesteDeploy - /home/fmsti/.rbenv/bin/rbenv exec bundle exec mina cdtest deploy - echo ”Deploy concluído com sucesso!”

Fonte: Elaborado pelo autor (2025).

4 RESULTADOS

Este capítulo apresenta a análise dos resultados alcançados pela metodologia empregada na pesquisa, desde a orquestração do ambiente até o *deploy* de uma modificação na aplicação. Os resultados obtidos demonstram o alcance de cada um dos objetivos específicos propostos, consolidando o desenvolvimento da esteira CI/CD.

4.1 Análise dos Objetivos Específicos

Para que fosse possível realizar a conexão entre a VPS e o repositório, a plataforma *Bitbucket* foi escolhida estrategicamente. Ela oferece integração nativa com ferramentas como *pipelines* e conexão direta com servidores, e por pertencer à empresa de tecnologia Atlassian, disponibiliza documentação abrangente e suporte técnico ao usuário. Outro fator decisivo foi a possibilidade de integração futura com outros serviços de DevOps. A escolha do plano *Standard* se deu por oferecer suporte preferencial, alta quantidade de ambientes de implantação e não possuir limite de usuários, o que abre a possibilidade de expansão da equipe.

Para configurar o *pipeline* da melhor forma, criamos um ambiente virtual com todas as dependências do projeto. Isso garantiu os ajustes corretos para a melhor integração do projeto. Os *steps* seguiram uma ordem já utilizada na FMS: primeiro, adequamos a escrita para que o código ficasse legível e compreensível a todos; em seguida, verificamos o correto funcionamento das funcionalidades implementadas; e, por fim, realizamos o *deploy* com o código já validado e aprovado.

Para adaptar o pipeline aos *steps* definidos, utilizamos a biblioteca *rubocop* no *step analyze*. Ela realiza a análise estática do código-fonte, garantindo sua qualidade ao identificar automaticamente erros, violações de estilo e boas práticas durante a escrita. Para o *step Tests*, empregamos a biblioteca *rubocop-rspec*, que verifica o estilo e a segurança dos arquivos de teste. Essa ferramenta pode ser acionada manualmente durante o desenvolvimento ou automaticamente no *pipeline*, validando a integridade dos testes e apontando inconsistências ou falhas de conformidade no código.

Para viabilizar a etapa de *deploy*, foram criados dois arquivos `shell script`. Um deles ficará armazenado no projeto e será executado durante o *pipeline*, tendo a função de acionar o segundo *script*. Este segundo arquivo, localizado na VPS, será responsável por atualizar o projeto e realizar o *deploy* da aplicação no servidor. Essa ação é possível graças à conexão SSH previamente configurada, que garante a comunicação segura entre o *pipeline* e o servidor remoto. Dessa forma, assegura-se que as

novas alterações sejam aplicadas e que a aplicação seja atualizada automaticamente no servidor.

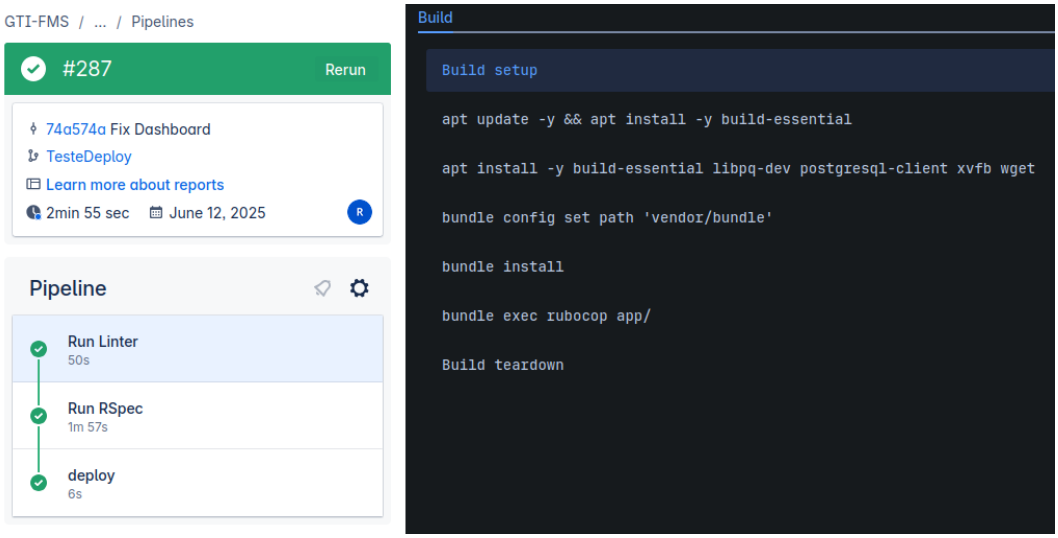
Para viabilizar a etapa de *deploy*, a solução encontrada foi criar dois arquivos `shell script`. O primeiro, armazenado no projeto, é executado durante o *pipeline* e aciona o segundo *script*. Este último, localizado na VPS, é responsável por atualizar o projeto e realizar o *deploy* da aplicação no servidor. Essa comunicação segura entre o *pipeline* e o servidor remoto é garantida pela conexão SSH previamente configurada, assegurando que as novas alterações sejam aplicadas e que a aplicação seja atualizada automaticamente no servidor.

4.2 Análise da Execução da Esteira CI/CD

Foi realizado um *commit* contendo uma alteração no *Dashboard* acompanhado da mensagem `Não existe aviso de campanha cadastrado`, com o objetivo de verificar o funcionamento da esteira de CI/CD, permitindo validar o fluxo automatizado de integração e *deploy*.

O *pipeline* inicia executando a etapa de *Linter*. Após a conclusão bem-sucedida, ele prossegue para os testes unitários. Com a finalização dessas etapas do CI, o *pipeline* realiza o *deploy* da aplicação na VPS, atualizando o ambiente com as novas alterações e validando o CD. Com todas as etapas concluídas, a aplicação apresentou a alteração de forma satisfatória, validando todo o processo. Todo o processo de validação pode ser visualizado no *Bitbucket* como mostra a imagem 4.

Figura 4 – Execução completa Pipeline



Fonte: Elaboração própria (2025)

Com todas as etapas concluídas, a aplicação apresentou a alteração proposta

de forma satisfatória, como apresentado na imagem 5, validando o funcionamento da esteira automatizada.

Figura 5 – Alteração

Fonte: Elaboração própria (2025)

5 CONCLUSÕES

Este trabalho de conclusão de curso atingiu seu objetivo geral de implementar uma esteira de CI/CD na FMS para automatizar o processo de *deploy* em um servidor VPS, utilizando conexão segura via SSH. Essa feito foi consolidada pelo desenvolvimento adequado de um pipeline CI/CD, que envolveu a análise estática do código e execução de testes unitários. Paralelamente, o projeto incluiu a automação do processo de *deploy*, viabilizado por uma conexão SSH entre VPS e repositório, proporcionando uma conexão segura. Finalmente, esteira de CI/CD foi implementada de forma satisfatória na FMS, a solução adotada mostrou-se eficaz e compatível com os requisitos de automação e segurança definidos, demonstrando-se como uma alternativa viável, eficiente e replicável para o cenário avaliado.

5.1 Conclusão

- A configuração da conexão SSH entre VPS e repositório foi realizada com êxito, permitindo que ocorresse uma comunicação e transferência de dados entre cliente servidor de maneira segura.
- O *pipeline* CI/CD foi configurado de forma a permitir a execução do projeto de forma virtual sem erros e execução de cada um dos *steps* definidos.
- A adaptação do *pipeline* para análise estática e testes unitários ocorreu de forma esperada, pois houve a execução dos mesmos durante seus respectivos *steps*.
- O funcionamento do *deploy* ocorreu de maneira satisfatória, sem apresentar erros e exibindo todas as alterações propostas.

5.2 Contribuições

Este trabalho busca contribuir de forma significativa para a área de DevOps, por meio do desenvolvimento de uma esteira de CI/CD. Espera-se que os resultados obtidos possam colaborar com o desenvolvimento de outros sistemas, inclusive aqueles construídos em linguagens de programação distintas, servindo como referência para a implementação de *pipelines* automatizados em ambientes controlados.

5.3 Limitações

A principal dificuldade enfrentada durante o desenvolvimento do projeto foi a liberação dos endereços IPs do Bitbucket. Isso se deve ao fato de que, para utilizar uma faixa restrita de IPs no *pipeline*, é necessário possuir um ambiente configurado em nuvem com suporte a *runners* dedicados ou *pipeline* hospedados, o que possibilita definir intervalos de IP específicos. Na ausência dessa configuração, seria necessário liberar mais de 300 endereços IP distintos, o que representaria uma potencial fragilidade para a infraestrutura de rede local, comprometendo a segurança do ambiente.

5.4 Trabalhos Futuros

Para trabalhos futuros, propõe-se a implementação de tecnologias de conteinerização, como *Docker*, e a adaptação da esteira de CI/CD para suportar múltiplas linguagens e bibliotecas. Além disso, sugere-se a análise e aplicação de diferentes métodos de *deploy* automatizado, possibilitando a ampliação do escopo e a modernização do processo de entrega contínua em ambientes diversos.

REFERÊNCIAS

ARACHCHI, S.; PERERA, I. Continuous integration and continuous delivery pipeline automation for agile software project management. In: IEEE. *2018 Moratuwa Engineering Research Conference (MERCon)*. [S.l.], 2018. p. 156–161. Citado na página 17.

DUVALL, P. M.; MATYAS, S.; GLOVER, A. *Continuous integration: improving software quality and reducing risk*. [S.l.]: Pearson Education, 2007. Citado na página 17.

GASPAR, L. Protocolo ssh: o que é e como funciona. *HostGator*, 2025. Disponível em: <https://www.hostgator.com.br/blog/o-que-e-protocolo-ssh/>. Citado na página 18.

GEA, C.; LASE, K. J. D.; SYAMSUDIN, M. Implementasi virtual private server untuk mini hosting. *Infact: International Journal of Computers*, v. 7, n. 01, p. 5–9, 2023. Citado na página 13.

HABBEMA, H. Desvendando ci/cd integração contínua e entrega contínua. *MEDIUM*, 2023. Disponível em: <https://medium.com/@habbema/desvendando-ci-cd-b56f515ddd20>. Citado na página 16.

HABBEMA, H. Ssh — secure shell. *MEDIUM*, 2023. Disponível em: <https://medium.com/@habbema/ssh-secure-shell-3b31a298d84e>. Citado na página 18.

HALL, T. Devops pipeline. *Atlassian*, Copyright © 2024 Atlassian, 2021. Disponível em: <https://www.atlassian.com/devops/devops-tools/devops-pipeline>. Citado na página 18.

LIMA, M. O que é vps e para que serve? *HostGator*, 2023. Disponível em: <https://www.hostgator.com.br/blog/servidor-vps-o-que-e-e-para-quem-e-indicado>. Citado na página 18.

SALES, V. A survey of devops concepts and challenges. *SYDLE*, 2025. Disponível em: <https://www.sydle.com/br/blog/ci-cd-678639b3e97eb8411eef7480>. Citado na página 16.

SANTOS, B. M. *Segurança em pipelines de CI/CD: análise de riscos, detecção de anomalias e notificações através de um chatbot inteligente*. Dissertação (B.S. thesis) — UNIVERSIDADE FEDERAL DE PERNAMBUCO, 2023. Citado na página 13.

SHAHIN, M. et al. Beyond continuous delivery: an empirical investigation of continuous deployment challenges. In: IEEE. *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. [S.l.], 2017. p. 111–120. Citado na página 17.

SHAHIN, M.; BABAR, M. A.; ZHU, L. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE access*, IEEE, v. 5, p. 3909–3943, 2017. Citado na página 17.

VIRTANEN, J. *Comparing Different CI/CD Pipelines*. Tese (Doutorado) — Häme University of Applied Sciences, 2021. Citado na página 17.

YLONEN, T.; LONVICK, C. *The secure shell (SSH) protocol architecture*. [S.l.], 2006. Citado na página 13.