Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Departamento de Engenharia Elétrica

Vitor Trindade Rocha Ribeiro

Relatório de Estágio Supervisionado

Campina Grande, Paraíba, Brasil
3 de outubro de 2024

Vitor Trindade Rocha Ribeiro

Relatório de Estágio Supervisionado

Relatório de Estágio Supervisionado submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Tecnologia da Informação e Comunicação

Orientador: Prof. Danilo Freire de Souza Santos

Campina Grande, Paraíba, Brasil
3 de outubro de 2024

Vitor Trindade Rocha Ribeiro

Relatório de Estágio Supervisionado

Relatório de Estágio Supervisionado submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Aprovado em: ___/___/___

Prof. Danilo Freire de Souza Santos
Orientador

Prof. Jaidilson Jó da Silva
Avaliador

Campina Grande, Paraíba, Brasil
3 de outubro de 2024

Agradecimentos

Minha profunda gratidão vai aos meus pais, Ana Paula e Flávio, cujo amor incondicional e dedicação incansável foram fundamentais para que eu chegasse até aqui. Todo o apoio, ensinamentos e cuidado ao longo da minha vida me guiaram e me deram a força necessária para conquistar este momento.

Agradeço aos meus irmãos, Vinicius e Laís, que, mesmo sem perceberem, me tornam uma pessoa melhor a cada dia. Vinicius, além de irmão, tive a sorte de compartilhar a mesma jornada profissional ao dividirmos a experiência na mesma empresa, o que tornou essa etapa ainda mais especial. Sem vocês, este momento não teria sido possível.

Minha gratidão também vai para minha amada Clara, que esteve ao meu lado nos últimos anos, tornando-se parte essencial da minha vida. Sempre pronta para me ouvir e aconselhar, sua presença e apoio foram fundamentais ao longo dessa jornada.

Sou grato a todos os amigos que fizeram parte da minha jornada: os que estiveram ao meu lado desde o início, compartilhando experiências e me inspirando com suas próprias vitórias, e os que conheci durante a graduação e durante o estágio, que tornaram esse caminho mais leve e significativo.

Meu sincero agradecimento ao Professor Danilo Freire de Souza Santos, meu orientador, pelo apoio e atenção contínuos ao longo deste trabalho. Estendo também minha gratidão aos colegas da deco.cx, especialmente a Hugo, Lucis e Candeia, por todos os ensinamentos e valiosa colaboração durante o estágio.

Agradeço à Universidade Federal de Campina Grande pelas oportunidades oferecidas e aos professores e funcionários do Departamento de Engenharia Elétrica, cujo apoio e dedicação foram essenciais para minha formação.

Lista de abreviaturas e siglas

UAEE	Unidade Acadêmica de Engenharia Elétrica
CEEI	Centro de Engenharia Elétrica e Informática
UFCG	Universidade Federal de Campina Grande
RBAC	*Role-Based Access Control*
IA	*Inteligência Artificial*
CMS	*Content Management System*
IoT	*Internet das Coisas*
DNS	*Domain Name System*
TI	*Tecnologia da Informação*

Lista de ilustrações

Sumário

1 Introdução

Este relatório descreve as atividades realizadas por Vitor Trindade Rocha Ribeiro, estudante do curso de Engenharia Elétrica da Universidade Federal de Campina Grande (UFCG), durante seu estágio na empresa deco.cx. O período do estágio foi de 1º de julho de 2024 a 20 de setembro de 2024, com uma carga horária semanal de 18 horas, totalizando 210 horas. O estágio foi realizado de forma remota, de segunda a sexta-feira, com todas as atividades desenvolvidas no setor de engenharia da deco.cx.

Durante o estágio, o estudante foi responsável pelo gerenciamento de ferramentas de orquestração de infraestrutura, aprimoramento da observabilidade, automação de diversas tarefas, além de suporte e manutenção em eventuais problemas com a infraestrutura.

O estágio foi supervisionado por Hugo Cavalcanti Cabral, engenheiro de software sênior na deco.cx, e orientado pelo professor Danilo Freire de Souza Santos.

1.1 Objetivos

Neste trabalho são detalhadas as atividades realizadas durante o estágio na deco.cx, concentrando-se na área de gestão e automação de infraestrutura orquestrada. As principais tecnologias e ferramentas utilizadas ao longo do estágio incluíram Kubernetes, Prometheus, Grafana, Helm, Bash, Pulumi, entre outras, para garantir uma infraestrutura eficiente e segura.

As principais atividades desenvolvidas durante o período de estágio foram:

- Configuração e gerenciamento de clusters Kubernetes;
- Monitoramento e gerenciamento de pods e serviços;
- Implementação de práticas de segurança, como RBAC (Role-Based Access Control) e uso de namespaces;
- Automação de deploys utilizando Helm charts;
- Implementação e gerenciamento de mecanismos de monitoramento e logging com Prometheus e Grafana;
- Análise de métricas de performance e otimização da infraestrutura;
- Configuração de alertas e notificações com PagerDuty;

- Automação de tarefas utilizando scripts Bash;
- Gerenciamento de infraestrutura como código com Pulumi;
- Suporte técnico e troubleshooting de problemas na infraestrutura;
- Colaboração com desenvolvedores e engenheiros para alinhar necessidades e objetivos técnicos.

1.2 Organização do Trabalho

Este relatório está organizado em seis capítulos. No **Capítulo 1** é apresentado a introdução, com uma descrição sucinta dos objetivos do trabalho e uma visão geral dos tópicos abordados ao longo do relatório. O **Capítulo 2** contém a apresentação da empresa, incluindo suas instalações e os principais projetos em que está envolvida. No **Capítulo 3**, são apresentados os fundamentos teóricos que sustentam o trabalho realizado. No **Capítulo 4** é apresentado as metodologias de desenvolvimento utilizadas no estágio. O **Capítulo 5** oferece uma descrição detalhada das atividades desenvolvidas durante o estágio. Por fim, no **Capítulo 6**, são discutidas as considerações finais, com os resultados alcançados e as lições aprendidas ao longo do período de estágio.

2 Local de Estágio

A deco.cx é uma plataforma frontend que preenche a lacuna entre construtores de sites básicos de low-code e configurações complexas de full-code, permitindo que empresas de todos os tamanhos criem experiências web de alto desempenho, totalmente personalizadas e com recursos nativos de IA. A plataforma, de código aberto, combina o melhor em estrutura, hospedagem, CMS, editor visual e editor de código.

A deco.cx foi fundada em agosto de 2022, no Rio de Janeiro, por Rafael, Guilherme e Luciano, com o objetivo de transformar o Brasil em uma referência em tecnologia. Luciano, ex-aluno de Ciência da Computação na UFCG, ajudou a dar vida a essa ideia. Em dezembro do mesmo ano, a empresa recebeu um investimento-anjo de 595 mil dólares e fechou seu primeiro contrato com a ZeeDog. Apenas dois anos depois, em 2024, a deco.cx já havia conquistado mais de 80 clientes, incluindo grandes marcas do mercado enterprise em todo o Brasil, e seu valor de mercado superava 10 milhões de dólares(DECO, 2024).

Figura 1 – Equipe da deco.cx no VTEX DAY 2024

A equipe da deco.cx é o principal alicerce da empresa, formada por profissionais de todo o país, que valorizam a diversidade de perspectivas e mantêm uma forte cultura

de responsabilidade e autonomia, se destacando pela proatividade e pela capacidade de assumir a responsabilidade pelos desafios, sempre comprometidos com a construção de um futuro extraordinário para a empresa. O perfil ideal da equipe é composto por pessoas que buscam inovação, têm a liberdade para identificar e resolver problemas, e se sentem confortáveis em colaborar e vislumbrar grandes objetivos. A cultura da equipe é pautada pela comunicação aberta, integridade e aprendizado contínuo, com uma abordagem que não teme erros e é guiada por uma visão de futuro promissor.

3 Fundamentação Teórica

Para embasar as atividades realizadas durante o estágio e fornecer um contexto técnico apropriado, este capítulo apresenta a fundamentação teórica que orientou o desenvolvimento das tarefas executadas.

3.1 Orquestração de Infraestrutura com Kubernetes

A orquestração de infraestrutura refere-se ao processo de automatizar o gerenciamento, coordenação e organização de sistemas e serviços, garantindo escalabilidade, alta disponibilidade e resiliência de aplicativos. Uma das principais ferramentas utilizadas para essa finalidade é o Kubernetes, uma plataforma open-source que facilita a execução e gerenciamento de contêineres em larga escala (KUBERNETES. . . , 2024a).

Kubernetes, originalmente desenvolvido pela Google, foi projetado para automatizar o processo de implantação, escalonamento e operação de aplicativos em contêineres. Ele oferece uma abstração sobre a infraestrutura física ou virtual, permitindo que os desenvolvedores e operadores lidem com contêineres de forma consistente, independentemente do ambiente subjacente (TURNBULL, 2014).

3.1.1 Componentes do Kubernetes

O Kubernetes é baseado em vários conceitos e componentes principais, entre os quais se destacam:

- **Clusters**: Um cluster Kubernetes consiste em um conjunto de máquinas, chamadas de nós, que executam os contêineres. O Kubernetes gerencia a distribuição de cargas de trabalho entre os nós, garantindo que os recursos sejam utilizados de maneira eficiente (BURNS et al., 2016).
- **Pods**: O menor e mais básico elemento no Kubernetes é o *pod*, que pode conter um ou mais contêineres. Os pods compartilham o mesmo espaço de rede e armazenamento e são projetados para serem efêmeros, com curta vida útil, facilitando o escalonamento e a recuperação em caso de falha (KUBERNETES. . . , 2024a).
- **Serviços**: Para garantir a comunicação entre pods e com o mundo externo, o Kubernetes oferece o conceito de *serviços*. Eles atuam como proxies que expõem os pods em execução, permitindo o balanceamento de carga e a comunicação entre diferentes componentes do sistema (KUBERNETES. . . , 2024a).

3.1.2 Segurança e Isolamento

No Kubernetes, a segurança é um fator crucial, e uma das práticas comuns é o uso de RBAC (Controle de Acesso Baseado em Funções). RBAC define permissões baseadas em funções atribuídas a usuários ou serviços, garantindo que as operações realizadas na infraestrutura sejam restritas conforme o nível de permissão de cada função (KUBERNETES. . . , 2024b).

Outro mecanismo de segurança implementado no Kubernetes é o uso de *namespaces*, que fornece isolamento lógico dentro de um cluster, permitindo que diferentes equipes ou projetos operem sem interferências, além de garantir limites de recursos e maior controle sobre as operações (NAMESPACES. . . , 2024).

3.1.3 Automação com Helm

Para facilitar o processo de deploy e gerenciamento de aplicativos em Kubernetes, uma ferramenta muito utilizada é o *Helm*. O Helm é um gerenciador de pacotes que usa *Helm charts*, que são templates pré-configurados para implantar e gerenciar serviços de forma consistente e escalável. Com Helm, a automação do processo de deploy e atualização de aplicativos se torna mais ágil e simplificada (HELM. . . , 2024).

3.1.4 Aplicações Multi-cluster, Multi-região e Cloud Agnostic

Uma das principais vantagens do Kubernetes é a capacidade de operar em ambientes multi-cluster e multi-região, oferecendo alta disponibilidade e recuperação de desastres. Além disso, o Kubernetes permite uma abordagem *cloud agnostic*, ou seja, ele pode ser executado em diferentes provedores de nuvem ou mesmo em data centers privados, garantindo flexibilidade e portabilidade (CLOUD. . . , 2024).

3.2 Computação em Borda

A computação em borda (*edge computing*) é uma arquitetura distribuída que move o processamento e o armazenamento de dados para mais próximo das fontes de dados, como dispositivos IoT, sensores e máquinas industriais, reduzindo a latência e o uso da largura de banda. Ao contrário dos modelos tradicionais de computação em nuvem, onde os dados são processados em data centers centrais, a computação em borda permite que os dados sejam analisados e processados localmente, proporcionando respostas em tempo real e melhorando a eficiência operacional (SATYANARAYANAN, 2017).

3.2.1 Benefícios da Computação em Borda

A computação em borda oferece várias vantagens para sistemas que exigem baixa latência, alta disponibilidade e maior segurança. Entre os principais benefícios estão:

- **Redução de latência**: Como o processamento é realizado mais próximo da fonte de dados, o tempo de resposta é significativamente reduzido, o que é essencial para aplicações em tempo real, como veículos autônomos, fábricas inteligentes e sistemas de saúde (SHI et al., 2016).
- **Eficiência no uso de banda**: Com a análise e o processamento de dados ocorrendo localmente, apenas informações relevantes são enviadas para a nuvem ou data centers, resultando em uma economia significativa de largura de banda (SATYANARAYANAN, 2019).
- **Melhoria na privacidade e segurança**: A execução de tarefas na borda pode minimizar o envio de dados sensíveis para servidores centrais, ajudando a proteger informações críticas de usuários ou dispositivos (YI; LI; LI, 2015).

3.2.2 Cloudflare e Computação em Borda

A Cloudflare, uma das principais plataformas de serviços de rede, tem desempenhado um papel significativo na implementação e evolução da computação em borda. Seu objetivo é fornecer uma infraestrutura distribuída para otimizar a entrega de conteúdos e serviços, aproximando o processamento de dados dos usuários finais. A computação em borda, nesse contexto, permite que os dados sejam processados e armazenados em servidores geograficamente distribuídos, conhecidos como *edge locations*, reduzindo a latência e melhorando a performance dos serviços (CLOUDFLARE, 2024c).

Na prática, a Cloudflare oferece uma rede de servidores de borda estrategicamente posicionados em diferentes locais ao redor do mundo. Esses servidores atuam como pontos de presença (PoPs), facilitando a entrega de conteúdo de forma eficiente e segura. Um exemplo de aplicação comum dessa infraestrutura é o uso em plataformas de e-commerce, onde os clientes são redirecionados automaticamente para o servidor de borda mais próximo. Isso garante tempos de resposta mais rápidos, melhorando a experiência do usuário e tornando o processo de compra mais eficiente (CLOUDFLARE, 2022).

Além da redução da latência, a Cloudflare também contribui para a segurança dos sistemas de borda, fornecendo proteção contra ataques DDoS, mitigando ameaças antes mesmo que elas cheguem ao servidor de origem (CLOUDFLARE, 2024b). Isso é particularmente importante em ambientes de computação em borda, onde dispositivos distribuídos são mais vulneráveis a ataques devido à sua proximidade com os usuários finais.

Outro aspecto crucial é a capacidade de computação em borda da Cloudflare, que permite a execução de funções diretamente nos servidores de borda por meio de sua plataforma *Cloudflare Workers*. Isso permite que aplicações sejam executadas mais próximas dos usuários, reduzindo a carga nos servidores centrais e permitindo uma escalabilidade mais eficiente (CLOUDFLARE, 2024a).

3.3 Observabilidade e Monitoramento

A observabilidade é um aspecto fundamental na gestão de infraestrutura moderna, especialmente em ambientes distribuídos e de alta complexidade. A observabilidade permite a coleta, monitoramento e análise de dados de sistemas em tempo real, proporcionando visibilidade sobre o desempenho, saúde e comportamento dos serviços. As três principais áreas de observabilidade são: logs, métricas e traces, sendo que ferramentas como Prometheus, Grafana e PagerDuty desempenham papéis essenciais nesse processo (FOWLER, 2020).

3.3.1 Prometheus: Coleta e Armazenamento de Métricas

O *Prometheus* é uma ferramenta open-source amplamente utilizada para monitoramento e alerta em sistemas distribuídos. Ele foi projetado para coletar, armazenar e consultar métricas de desempenho e de estado em tempo real, sendo ideal para ambientes baseados em microserviços e contêineres, como Kubernetes (PROMETHEUS. . . , 2024).

Prometheus coleta métricas em intervalos regulares de endpoints de sistemas ou serviços através de *pull requests*. As métricas são armazenadas em uma base de dados local em formato de séries temporais, o que facilita a análise histórica e a geração de alertas baseados em limites predefinidos (TURNBULL, 2018). Sua integração com o Kubernetes permite o monitoramento detalhado de pods, serviços e nós, fornecendo uma visão abrangente da infraestrutura.

3.3.2 Grafana: Visualização de Dados e Dashboards

O *Grafana* é uma plataforma open-source de visualização de dados que se integra perfeitamente com Prometheus e outras fontes de dados, permitindo a criação de dashboards interativos e personalizados para monitoramento em tempo real (GRAFANA. . . , 2024). Uma das grandes vantagens do Grafana é sua capacidade de unificar dados de diferentes fontes (métricas, logs, traces) em um único painel, proporcionando uma visão abrangente e detalhada do estado da infraestrutura.

Com o Grafana, é possível criar alertas visuais e gatilhos de notificação baseados em métricas específicas. Além disso, sua interface amigável permite a construção rápida

de dashboards customizados, onde administradores podem monitorar em tempo real o consumo de recursos, tempo de resposta de aplicações, latência de serviços, entre outros indicadores de desempenho (GRAFANA. . . , 2024).

A combinação de Prometheus e Grafana é poderosa: enquanto Prometheus coleta e armazena métricas, o Grafana permite que essas métricas sejam visualizadas e analisadas de maneira eficiente. Isso facilita a detecção proativa de problemas, como consumo excessivo de CPU, quedas de serviços ou gargalos de rede.

3.3.3 PagerDuty: Gerenciamento de Incidentes e Alertas

O *PagerDuty* é uma plataforma voltada para o gerenciamento de incidentes e resposta a alertas. Ele se integra com Prometheus e Grafana para fornecer notificações imediatas em caso de falhas, incidentes críticos ou violações de limites de performance (PAGERDUTY. . . , 2024a). Quando uma métrica monitorada por Prometheus atinge um limite definido (por exemplo, uso de CPU acima de 90 porcento por um período de tempo), o PagerDuty é acionado para notificar a equipe responsável por meio de canais como SMS, e-mail ou chamadas telefônicas, garantindo uma resposta rápida e organizada (PAGERDUTY. . . , 2024c).

O PagerDuty permite a criação de escalas de plantão, onde equipes podem ser acionadas de acordo com a criticidade do incidente. Além disso, ele oferece uma estrutura de escalonamento de alertas, onde, se o primeiro responsável não responder, o incidente é automaticamente escalado para o próximo na linha de plantão (PAGERDUTY. . . , 2024b). Isso garante que os incidentes sejam resolvidos rapidamente e com o mínimo de impacto para o negócio.

4 Métodos de Desenvolvimento

Para o desenvolvimento das atividades durante o estágio na deco.cx, foram utilizados diversos métodos, ferramentas e rituais que auxiliaram na organização e no acompanhamento das demandas. As principais ferramentas adotadas pela empresa foram o *Discord* e o *Notion*, que desempenharam papéis fundamentais no fluxo de trabalho e na comunicação interna.

4.1 Ferramentas Utilizadas

O *Notion* foi a principal ferramenta de gestão de atividades. Todas as tarefas eram organizadas em um *backlog*, com cada tarefa atrelada a um responsável específico. Além disso, o *Notion* permitia o acompanhamento de *milestones*, metas e objetivos maiores, garantindo que todos os prazos fossem monitorados e que houvesse clareza sobre o andamento dos projetos. Essa estrutura facilitava o planejamento e a execução das atividades, além de permitir uma visão clara das prioridades e entregas.

O *Discord* funcionava como o principal canal de comunicação da empresa. Através dele, as equipes mantinham um contato constante, seja nas salas de voz para discussões rápidas e alinhamentos diários, ou nas salas de texto para trocas de informações e atualizações. Uma prática comum no *Discord* era o uso de *workthreads*, que consistiam em salas específicas dentro de um fórum para tratar de tópicos específicos ou resolver problemas pontuais. Essa abordagem favorecia a colaboração e o acompanhamento das atividades de maneira ágil e eficiente.

4.2 Rituais de Desenvolvimento

Na deco.cx, além das ferramentas, a equipe seguia um conjunto de rituais semanais que auxiliavam no acompanhamento e desenvolvimento das demandas de forma organizada e transparente.

4.2.1 Week Start e Priorization

Toda segunda-feira, duas reuniões importantes aconteciam: a *Week Start* e a *Priorization*. A *Week Start* começava com o momento *Good News*, onde cada membro da equipe tinha a oportunidade de compartilhar boas notícias ou acontecimentos pessoais, criando um ambiente de descontração e integração. Em seguida, acontecia o *Customer*

or Team Headline, onde cada pessoa antecipava eventos importantes da semana, como reuniões, folgas ou prazos importantes, alinhando expectativas para a semana.

Logo após, ocorria a reunião de *Priorization*, cujo foco era organizar as tarefas da semana. Com o auxílio do *Notion*, a equipe revisava o andamento das demandas, verificava o que havia sido concluído, o que precisava de mais tempo e quais seriam as próximas prioridades. Esse ritual era essencial para manter o alinhamento entre os times e garantir que todos estivessem cientes das responsabilidades e prazos.

4.2.2 Product Critique

Às terças-feiras, acontecia a *Product Critique*, uma reunião voltada para a análise crítica de algum aspecto da empresa, como a interface de uma tela, o funcionamento de uma parte da infraestrutura, ou qualquer outro ponto relevante. O objetivo não era criticar negativamente, mas sim desenvolver um senso crítico construtivo, buscando sempre melhorias que estivessem alinhadas com os objetivos estratégicos da empresa. Essa reunião fomentava uma cultura de melhoria contínua, em que todos podiam contribuir com sugestões e críticas construtivas.

4.2.3 Week End

A semana de trabalho na deco.cx terminava com a *Week End*, realizada nas sextas-feiras. Essa reunião começava com o *Conversation Cleanup*, um espaço para que qualquer pessoa pudesse expressar questões que estivessem gerando desconforto, ansiedade ou preocupação, com o intuito de encontrar soluções e resolver conflitos.

Em seguida, havia o momento de *Low Lights*, no qual os membros da equipe podiam compartilhar os problemas ou dificuldades enfrentados durante a semana, promovendo um ambiente de transparência e aprendizado. A reunião terminava com o *Results and Progress*, onde cada um apresentava seus resultados e progressos, oferecendo uma visão ampla sobre o que foi realizado e o status geral da empresa. Esse momento era fundamental para garantir que todos tivessem clareza sobre o andamento dos projetos e os objetivos atingidos.

4.3 Impacto dos Métodos de Desenvolvimento

A adoção dessas ferramentas e rituais ajudou a promover um ambiente de trabalho colaborativo, organizado e transparente. O uso do *Notion* e do *Discord* possibilitou a gestão eficiente das tarefas e a comunicação rápida e eficaz entre os membros da equipe. Já os rituais semanais garantiram o alinhamento constante, a reflexão crítica sobre os

processos da empresa e a transparência no compartilhamento de resultados, contribuindo para um ambiente de trabalho mais produtivo e harmonioso.

5 Atividades Realizadas

Durante o estágio na deco.cx, o aluno foi responsável por diversas atividades relacionadas ao gerenciamento de infraestrutura orquestrada, automação de processos e monitoramento de sistemas. As ações realizadas foram fundamentais para otimizar o desempenho da plataforma, melhorar a eficiência operacional e garantir a escalabilidade e segurança dos serviços. No entanto, devido a termos de confidencialidade assinados pelo aluno, este capítulo não contém ilustrações relacionadas às atividades descritas. A seguir, são detalhadas as principais atividades desenvolvidas ao longo do estágio.

5.1 Automação e Escalabilidade da Infraestrutura

Uma das principais iniciativas foi a implementação da *Action de Event Schedule*, que automatizou o escalonamento da infraestrutura Kubernetes para grandes eventos, como a Black Friday. Com essa ação, tornou-se possível definir datas de eventos, e a infraestrutura se preparava automaticamente para lidar com picos de demanda, ajustando a capacidade dos serviços sem necessidade de intervenção manual. Essa solução garantiu que a plataforma mantivesse um desempenho estável durante momentos críticos de tráfego elevado.

Outra atividade importante foi o desenvolvimento da *Action de Redeploy*, que simplificou significativamente o processo de atualização dos sites dos clientes. Anteriormente, a execução de um redeploy na infraestrutura Kubernetes exigia conhecimento técnico especializado, mas com essa automatização, agora é possível realizar essa operação com um simples clique, reduzindo a complexidade operacional e agilizando a implementação de mudanças no ambiente.

Com o objetivo de facilitar o ajuste da escalabilidade dos serviços, foi implementado a *Action de Scaling*. Esta ação permitiu que o dimensionamento dos sites hospedados no Kubernetes fosse ajustado de maneira automatizada, eliminando a necessidade de acessar arquivos do cluster ou de utilizar ferramentas específicas. A automação desse processo aumentou a eficiência e melhorou a gestão de recursos da infraestrutura.

5.2 Segurança e Controle de Acesso

Visando aumentar a segurança e o controle de acesso no ambiente Kubernetes, foi desenvolvido um sistema de *Cargos RBAC* (Role-Based Access Control). Inicialmente, apenas um grupo restrito de usuários tinha acesso ao cluster, por questões de segurança.

Com a implementação do RBAC, foi criado uma hierarquia de permissões que possibilitou que todos os membros da empresa tivessem o nível de acesso adequado, conforme suas funções, sem comprometer a integridade do sistema. Além disso, o processo de login e acesso ao cluster foi simplificado, facilitando o uso da infraestrutura por colaboradores menos familiarizados com tecnologias específicas.

5.3 Migração de Infraestrutura

Durante o estágio, o aluno participou de várias operações de migração de sites, transferindo aplicações de clientes da infraestrutura do *denodeploy* para a infraestrutura própria da deco.cx. Esse processo envolveu a configuração tanto do Kubernetes quanto do Cloudflare para gerenciar DNS e proxy reverso. As migrações incluíram desde pequenos sites até grandes e-commerces, garantindo uma transição suave e minimizando o tempo de inatividade dos serviços.

5.4 Otimização de Performance

Devido ao grande volume de usuários, o cluster Kubernetes da deco.cx acumula milhares de namespaces, muitos dos quais se tornam inativos com o tempo, o que pode prejudicar o desempenho geral do sistema. Para resolver esse problema, foi desenvolvido um *script de limpeza de rotas* utilizando `Bash`. Esse script automatizou a identificação e remoção de namespaces inativos, melhorando a eficiência e desempenho da infraestrutura.

Outro projeto crucial desenvolvido foi o *Min 0*, que tinha como objetivo otimizar o uso de recursos durante o processo de deploy. A solução escalava automaticamente os antigos deployments para zero pods assim que um novo deployment era realizado. Antes dessa automatização, a prática comum era apagar manualmente os deployments antigos, o que demandava muito tempo e esforço da equipe de engenharia. Com essa otimização, o uso de recursos foi significativamente melhorado.

5.5 Desenvolvimento de Ferramentas Internas

Durante o estágio, foi iniciado o desenvolvimento do *Platform Dashboard*, um painel de controle integrado ao admin da deco.cx. Esse dashboard permitiu que a equipe gerenciasse de forma centralizada as configurações dos sites hospedados na nossa infraestrutura, bem como realizasse alterações diretamente no cluster Kubernetes através de uma interface web. O desenvolvimento dessa ferramenta simplificou a gestão da infraestrutura e permitiu que ajustes fossem feitos de maneira rápida e eficiente, sem a necessidade de acessar diretamente o Kubernetes.

5.6 Monitoramento e Suporte

Durante todo o período do estágio, o estagiário foi responsável por monitorar a infraestrutura utilizando ferramentas como *Grafana* e *Prometheus*. Essas ferramentas forneciam métricas em tempo real sobre o desempenho dos serviços e permitiam a detecção proativa de problemas. Além disso, o aluno atuou no suporte oncall, sendo notificado via *PagerDuty* sempre que incidentes críticos ocorriam. Quando problemas eram detectados, era necessário o auxilio no processo de depuração e resolução, garantindo que o impacto sobre os clientes fosse minimizado e que a infraestrutura se mantivesse estável.

6 Considerações Finais

Este relatório apresenta as atividades desenvolvidas durante o estágio na deco.cx e destaca a importância dessa experiência para minha formação acadêmica e profissional. Durante o período na empresa, foi possível aplicar os conhecimentos adquiridos ao longo da graduação em Engenharia Elétrica e adquiridos por conta própria, em um ambiente prático e desafiador, voltado para a gestão de infraestrutura orquestrada e automação de processos.

As atividades desempenhadas aprofundaram meus conhecimentos em áreas cruciais como Kubernetes, automação com Helm, monitoramento com Prometheus e Grafana, e ferramentas de gerenciamento de incidentes como PagerDuty. O trabalho com a orquestração de infraestrutura em ambientes distribuídos, multi-cluster e cloud agnostic, bem como a configuração de soluções de computação em borda, foi essencial para entender o funcionamento e a complexidade de arquiteturas modernas de TI.

Apesar disso, é importante ressaltar que a Universidade Federal de Campina Grande (UFCG) não ofereceu uma base profunda em disciplinas relacionadas a arquiteturas distribuídas, Kubernetes e ferramentas modernas de observabilidade. Por conta disso, foi necessário um estudo intenso por iniciativa própria para alcançar o nível de conhecimento exigido pelo estágio. Essa lacuna no currículo acadêmico demandou um esforço considerável para me atualizar sobre esses temas, que são fundamentais para a atuação no mercado de tecnologia atual.

Além disso, colaborar diretamente com engenheiros e desenvolvedores da deco.cx, em um ambiente dinâmico e inovador, foi fundamental para aprimorar tanto habilidades técnicas quanto a capacidade de trabalhar em equipe. O desenvolvimento de soluções como a automação de deploys e escalabilidade, criação de cargos RBAC, migração de infraestruturas, e o desenvolvimento de dashboards para monitoramento, reforçou minha capacidade de solucionar problemas de forma eficiente e criativa.

O estágio também proporcionou um contato direto com o gerenciamento de infraestrutura como código e automação de tarefas com scripts, consolidando a compreensão de práticas modernas na área de DevOps. A experiência prática em ambientes de produção, aliada à constante necessidade de monitorar e otimizar a infraestrutura, contribuiu significativamente para meu crescimento profissional, oferecendo uma visão clara sobre os desafios e responsabilidades que envolvem o gerenciamento de infraestrutura em larga escala.

Em suma, o estágio na deco.cx não só consolidou o conhecimento teórico aprendido durante a graduação, mas também me preparou para enfrentar problemas reais, aplicar

soluções tecnológicas de ponta e colaborar de forma eficiente em equipes multidisciplinares. Essa experiência foi crucial para o desenvolvimento das competências necessárias para atuar em ambientes tecnológicos complexos e dinâmicos.

Referências

BURNS, B. et al. Borg, omega, and kubernetes. In: *Communications of the ACM*. [S.l.]: ACM, 2016. v. 59, n. 5, p. 50–57. 5

CLOUD Native Kubernetes. 2024. Disponível em: <https://www.cncf.io/projects/kubernetes/>. 6

CLOUDFLARE. *How Cloudflare uses Edge Locations to Improve Website Performance*. 2022. Disponível em: <https://www.cloudflare.com/performance/>. 7

CLOUDFLARE. *Cloudflare Workers: Serverless Computing at the Edge*. 2024. Disponível em: <https://workers.cloudflare.com/>. 8

CLOUDFLARE. *DDoS Protection for Edge Computing*. 2024. Disponível em: <https://www.cloudflare.com/ddos/>. 7

CLOUDFLARE. *What is Edge Computing?* 2024. Disponível em: <https://www.cloudflare.com/learning/serverless/what-is-edge-computing/>. 7

DECO. 2024. Disponível em: <https://deco.cx/>. 3

FOWLER, M. *Observability: Why It Matters and How to Approach It*. 2020. Disponível em: <https://martinfowler.com/articles/observability.html>. 8

GRAFANA Documentation. 2024. Disponível em: <https://grafana.com/docs/>. 8, 9

HELM - The Kubernetes Package Manager. 2024. Disponível em: <https://helm.sh/>. 6

KUBERNETES Documentation. 2024. Disponível em: <https://kubernetes.io/docs/>. 5

KUBERNETES Security. 2024. Disponível em: <https://kubernetes.io/docs/concepts/security/>. 6

NAMESPACES in Kubernetes. 2024. Disponível em: <https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/>. 6

PAGERDUTY Documentation. 2024. Disponível em: <https://support.pagerduty.com/docs>. 9

PAGERDUTY Escalation Policies. 2024. Disponível em: <https://support.pagerduty.com/docs/escalation-policies>. 9

PAGERDUTY: Incident Response. 2024. Disponível em: <https://www.pagerduty.com/use-cases/incident-response/>. 9

PROMETHEUS Documentation. 2024. Disponível em: <https://prometheus.io/docs/>. 8

SATYANARAYANAN, M. The emergence of edge computing. *IEEE Computer*, v. 50, n. 1, p. 30–39, 2017. 6

SATYANARAYANAN, M. The emergence of edge computing. In: *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems*. [S.l.: s.n.], 2019. p. 73–74. 7

SHI, W. et al. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, v. 3, n. 5, p. 637–646, 2016. 7

TURNBULL, J. *The Docker Book: Containerization is the new virtualization*. USA: James Turnbull Publishing, 2014. 5

TURNBULL, J. *The Prometheus Monitoring Book*. USA: Turnbull Press, 2018. 8

YI, S.; LI, C.; LI, Q. A survey of fog computing: Concepts, applications, and issues. *Proceedings of the 2015 Workshop on Mobile Big Data*, p. 37–42, 2015. 7