Plataforma de Serviços de Rede Baseada em containers

Daniel Taveira - a44319

Trabalho realizado sob a orientação de

Prof. Nuno Rodrigues

Prof. Eduardo Costa

Licenciatura em Engenharia Informática

2025-2026

Plataforma de Serviços de Rede Baseada em containers

Relatório da UC de Projeto

Licenciatura em Engenharia Informática

Escola Superior de Tecnologia e Gestão

Daniel Taveira - a44319

2025-2026

A Escola Superior de Tecnologia e de Gestão não se responsabiliza pelas opiniões expressas neste relatório.

Declaro que o trabalho descrito neste relatório é da minha autoria e é da minha vontade que o mesmo seja submetido a avaliação.

Daniel Taveira - a44319

Dedicatória

(Facultativo) Dedico este trabalho a ...

Agradecimentos

(Facultativo) Agradeço a ...

Resumo

O presente projeto tem como objetivo principal a conceção e implementação de uma plataforma de serviços de rede gerida de forma reprodutível, recorrendo exclusivamente a ferramentas *open source*. A infraestrutura desenvolvida engloba serviços fundamentais para uma rede de pequena a média dimensão, tais como servidor DNS, *Reverse Proxy* HTTPS, serviços *web*, e-mail, servidor de ficheiros e VPN.

Em contraste com abordagens monolíticas tradicionais, todos os serviços são implementados em *containers* (Docker/Podman) através de configuração declarativa, garantindo o versionamento e a reprodutibilidade do ambiente. O trabalho foca-se na aplicação rigorosa de boas práticas de segurança, incluindo o princípio do mínimo privilégio, *hardening* de serviços e encriptação TLS. Adicionalmente, o projeto integra mecanismos avançados de observabilidade (para métricas e *logs* centralizados) e a orquestração de *deployments* através de ferramentas como Ansible e *pipelines* de CI/CD. O resultado final demonstra como um pequeno *data center* pode ser gerido de forma automatizada, ágil e resiliente a falhas.

Palavras-chave: *Containers*, Orquestração, Serviços de Rede, CI/CD, Observabilidade, *Open Source*.

Abstract

The main objective of this project is the design and implementation of a network services platform managed in a reproducible way, using exclusively open-source tools. The developed infrastructure encompasses fundamental services for a small to medium-sized network, such as a DNS server, HTTPS Reverse Proxy, web services, e-mail, file server, and a VPN.

In contrast to traditional monolithic approaches, all services are implemented in containers (Docker/Podman) through declarative configuration, ensuring environment versioning and reproducibility. The work focuses on the strict application of security best practices, including the principle of least privilege, service hardening, and TLS encryption. Additionally, the project integrates advanced observability mechanisms (for metrics and centralized logs) and deployment orchestration using tools like Ansible and CI/CD pipelines. The final result demonstrates how a small data center can be managed in an automated, agile, and fault-resilient manner.

Keywords: Containers, Orchestration, Network Services, CI/CD, Observability, Open Source.

Índice

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A administração de sistemas e redes tem sofrido uma evolução drástica nos últimos anos. A necessidade de implementar e gerir serviços de forma rápida, segura e sem interrupções impulsionou a adoção de tecnologias de *containerização* e práticas de automação de infraestruturas.

1.1 Contextualização e Motivação

Numa infraestrutura tecnológica de pequena ou média dimensão, a gestão manual de múltiplos serviços (como DNS, *web*, *proxies* e VPNs) torna-se insustentável a longo prazo, sendo propensa a erros humanos e dificultando a recuperação em caso de falhas. A adoção do paradigma de "Infraestrutura como Código"(IaC) e de arquiteturas baseadas em *containers* permite mitigar estes problemas, oferecendo ambientes isolados, padronizados e facilmente replicáveis.

1.2 Objetivos

O objetivo central deste projeto é construir uma plataforma de serviços de rede baseada em *containers* e orquestração leve, focada em boas práticas de segurança, CI/CD (*Continuous Integration / Continuous Deployment*) e observabilidade.

De forma mais específica, pretende-se:

- Implementar serviços típicos de rede (DNS, *Reverse Proxy* HTTPS, *Web*, E-mail, Servidor de Ficheiros e VPN) em *containers* (Docker/Podman).
- Demonstrar a gestão de um pequeno *data center* de forma totalmente reprodutível utilizando apenas ferramentas *open source*.
- Integrar mecanismos de observabilidade para a recolha de métricas e *logs* centralizados.
- Automatizar a configuração e o *deploy* com recurso a Ansible e a uma *pipeline* simples de CI/CD.

1.3 Estrutura do Documento

O presente relatório encontra-se organizado da seguinte forma: o Capítulo 2 apresenta o Estado da Arte das tecnologias envolvidas; o Capítulo 3 detalha o levantamento das necessidades funcionais e a escolha do ambiente base; o Capítulo 4 descreve a arquitetura e implementação da plataforma; o Capítulo 5 aborda os testes e resultados obtidos; por fim, o Capítulo 6 sintetiza as conclusões do trabalho.

Capítulo 2

Estado da Arte

A transição de infraestruturas baseadas em máquinas virtuais monolíticas para arquiteturas orientadas a *containers* revolucionou a administração de sistemas. Este capítulo explora as tecnologias e os paradigmas atuais que suportam a criação de plataformas de serviços de rede ágeis, seguras e observáveis.

2.1 Containerização e Orquestração Leve

Tradicionalmente, a segregação de serviços exigia a alocação de máquinas virtuais independentes, o que resultava num elevado *overhead* de recursos e numa gestão complexa. Atualmente, motores de *containerização* como o Docker e o Podman tornaram-se o padrão da indústria. Estas ferramentas permitem encapsular uma aplicação e todas as suas dependências num ambiente isolado, que partilha o *kernel* do sistema operativo *host*.

O Podman, em particular, destaca-se pela sua arquitetura *daemonless* e pela capacidade de executar *containers* sem privilégios de *root*, indo ao encontro das melhores práticas de segurança (o princípio do mínimo privilégio). Para a orquestração de infraestruturas de pequena a média dimensão, ferramentas declarativas como o Docker Compose oferecem a "orquestração leve"ideal. Permitem definir infraestruturas complexas como código (IaC - *Infrastructure as Code*) de forma totalmente reprodutível e facilmente versionável.

2.2 Observabilidade Centralizada

Manter um conjunto de serviços distribuídos exige mais do que uma simples monitorização; exige observabilidade. O estado da arte baseia-se em *stacks open source* complementares:

- **Métricas:** O Prometheus atua como o motor principal de recolha de métricas em formato *time-series*, trabalhando em conjunto com o Grafana para a visualização avançada de dados através de *dashboards* dinâmicos.
- **Logs:** Soluções como a *stack* ELK (Elasticsearch, Logstash, Kibana) ou o Loki (otimizado pela Grafana Labs para ambientes de *containers*) permitem agregar *logs* de múltiplos serviços num único ponto central. Isto facilita substancialmente o processo de *troubleshooting*.

2.3 Segurança e CI/CD em Ambientes Locais

A segurança *by design* é um requisito fundamental na gestão de redes modernas. Isto inclui o uso sistemático de *Reverse Proxies* (como o Nginx, Traefik ou Caddy), que gerem a terminação TLS/SSL de forma automática, e o *hardening* de imagens base.

A adoção de *pipelines* de CI/CD (Integração e Entrega Contínuas) deixou de ser um processo exclusivo do desenvolvimento de *software* e passou a integrar a gestão de infraestruturas (GitOps). A utilização de repositórios Git acoplados a *runners* locais permite que qualquer alteração na infraestrutura (como uma atualização de versão do servidor DNS ou a adição de um novo serviço Web) seja testada antes do *deploy* para o ambiente de produção. Esta prática reduz o erro humano e garante um *rollback* fiável em caso de falha dos serviços.

Capítulo 3

Levantamento das Necessidades Funcionais

Com base no guião do projeto proposto, o objetivo central consiste na criação de um "data center"de serviços de rede que possa ser gerido de forma reprodutível, recorrendo exclusivamente a ferramentas *open source*. Para atingir este objetivo, as necessidades do projeto dividem-se em duas categorias principais: os serviços a disponibilizar (requisitos funcionais) e os requisitos técnicos e de arquitetura (requisitos não-funcionais).

3.1 Serviços de Rede a Disponibilizar

A infraestrutura deve contemplar um conjunto de serviços típicos de uma rede de pequena/média dimensão. Os serviços a implementar são os seguintes:

- **Servidor DNS:** Implementação de um servidor com capacidade recursiva e autoritativa para a resolução de nomes na rede interna.
- **Reverse Proxy HTTPS:** Criação de um ponto de entrada único e seguro para o encaminhamento de tráfego para os diversos serviços internos.
- **Serviços Web e de E-mail:** Alojamento de páginas *web* e disponibilização de um sistema de gestão básica de correio eletrónico.

- **Servidor de Ficheiros:** Sistema centralizado para o armazenamento e a partilha de ficheiros na rede.

- **VPN (*Virtual Private Network*):** Implementação de um serviço que permita o acesso remoto e seguro à infraestrutura interna de *containers*.

3.2 Requisitos Técnicos e de Arquitetura

Para além do correto funcionamento dos serviços, a plataforma tem de respeitar rigorosos critérios de implementação e gestão:

- **Containerização e Configuração Declarativa:** Todos os serviços têm de ser executados em *containers* (Docker ou Podman). A configuração deve ser feita de forma declarativa (utilizando ficheiros como o `docker-compose.yml`), o que garante a sua reprodutibilidade e permite o versionamento.

- **Observabilidade e Monitorização:** É obrigatória a integração de uma *stack open source* (por exemplo, Prometheus, Grafana, Loki ou ELK/EFK) para a recolha contínua de métricas, gestão de *logs* centralizados e configuração de alertas básicos (como latência, erros 5xx ou saturação de CPU/RAM).

- **Segurança (*Hardening*):** O projeto exige a aplicação do princípio do mínimo privilégio nos *containers*, gestão segura e isolada de segredos, encriptação obrigatória de tráfego (TLS) e uma política de atualização automatizada das imagens dos *containers*.

- **Pipeline de CI/CD:** Deve ser implementada uma *pipeline* simples (utilizando, por exemplo, o Git em conjunto com um *runner* local) para testar automaticamente as alterações de configuração e assegurar um *deploy* controlado para o ambiente de produção do laboratório.

3.3 Ambiente Base: Virtualização e Sistema Operativo

Para suportar a plataforma e garantir o isolamento do *host* físico, a infraestrutura base será implementada em ambiente virtualizado.

- **Virtualização (VMware):** A plataforma escolhida para o *hypervisor* local é o VMware (Workstation/Player). Esta escolha justifica-se pela sua estabilidade, excelente gestão de redes virtuais (Bridged/NAT) necessárias para simular o tráfego externo/interno e eficiência na alocação de recursos.
- **Sistema Operativo (*Guest OS*):** Para o sistema operativo que irá alojar os *containers*, optou-se por uma distribuição Linux orientada a servidores, nomeadamente o **Ubuntu Server 24.04 LTS** (ou alternativamente o **Debian 12**). Estas distribuições foram selecionadas por possuírem vasto suporte da comunidade, excelente compatibilidade nativa com o Docker/Podman e com o Ansible, e por não incluírem interfaces gráficas (*headless*), o que minimiza a superfície de ataque e o consumo de recursos.

Capítulo 4

Desenvolvimento/Implementação

Neste capítulo é descrito o trabalho de implementação, salientando os pontos mais relevantes da mesma, dificuldades encontradas ou soluções técnicas inovadoras desenvolvidas ou aplicadas. Em particular, se foi usado código desenvolvido por terceiros (por exemplo, código *open-source*), deve ser facilmente distinguível quais as funcionalidades originais do mesmo e o que foi necessário implementar para obter as funcionalidades desejadas.

Capítulo 5

Testes/Avaliação/Discussão

Este capítulo apresenta os testes realizados para verificar que o projeto desenvolvido cumpre os objetivos assumidos e resolve, de facto, o problema descrito na Análise/Modelação.

Para uma melhor compreensão, os resultados de cada teste devem ser precedidos de uma descrição, mesmo que resumida, do teste realizado e dos resultados esperados.

Os resultados do trabalho são comentados, acrescentando-lhe valor:

- O que é que se pode inferir ou conjeturar dos resultados obtidos?
- O que poderia/deveria ter sido feito de forma diferente?
- Onde se foi além dos objetivos iniciais?
- Quais os objetivos que ficaram por cumprir, e porquê ?

Capítulo 6

Conclusões

As conclusões devem sintetizar e proporcionar uma perspetiva unificadora ao trabalho efetuado. Poderá ser feita uma breve referência a trabalhos de outros com semelhanças ao efetuado e ao conhecimento que resultou do trabalho efetuado, bem como sugestões de trabalho futuro. A coerência do documento implica que as conclusões devem ser coerentes com as ideias expostas na introdução.

Apêndice A

Proposta Original do Projeto

Curso de Licenciatura em Engenharia Informática

Projeto 3º Ano - Ano letivo de 2016/2017

<Título do projeto>

Orientador: <Nome do orientador>

Coorientador: <Nome do coorientador>

1 Objetivo

<Objetivo do projeto>

2 Detalhes

<Detalhes que julguem ser necessários>

3 Metodologia de trabalho

<Eventual metodologia de trabalho>

Dimensão da equipa:

Recursos necessários:

Apêndice B

Outro(s) Apêndice(s)

Listagens de código fonte, texto/imagens produzidos por testes complementares, etc.