Secure Exposure of REST APIs in Homelab Environments using Cloudflare Tunnel

Ulisses Thorwald Moraes Guedes[1], **Eduardo Cabezudo Vilhalba**[1], **Danielly Cristina do Carmo Neves**[1], **Silvio E. Quincozes**[1]

[1] Universidade Federal do Pampa (UNIPAMPA)
Alegrete – RS – Brasil

{eduardovilhalba, ulissesguedes, daniellyneves}.aluno@unipampa.edu.br

silvioquincozes@unipampa.edu.br

Abstract. *The rising cost of cloud platforms has motivated students, hobbyists, and small teams to host low-demand applications in homelabs. Although this approach reduces costs and supports practical learning, exposing services from residential networks introduces security and availability risks. This paper presents a case study on securely exposing a homelab-hosted REST API using secure backend practices, a Dokku-managed Linux virtual machine, PostgreSQL, SSH hardening, and Cloudflare Tunnel. The results indicate that this architecture is a low-cost and replicable alternative for small applications, reducing direct network exposure while preserving encrypted access, centralized traffic handling, and a smaller attack surface.*

1. Introduction

Cloud platforms provide scalability, availability, and managed security, but their recurring costs may be prohibitive for students, small teams, non-profit projects, and applications with modest computational demands. Homelabs offer a low-cost environment for learning, experimentation, and deployment on residential devices or reused servers [Brush et al. 2012]. However, residential networks rely on consumer-grade routers, limited observability, non-redundant connectivity, and constrained physical infrastructure. Consequently, exposing a homelab-hosted service changes the system's security assumptions and makes network exposure a central concern.

The main challenge addressed in this work is how to expose a small backend application to the Internet without directly opening ports in a residential router. Direct exposure increases the attack surface of the local network and may require the developer to manually manage firewall rules, public addressing, TLS certificates, access control, and traffic filtering. These concerns are especially relevant for REST APIs, since APIs expose endpoints that handle application resources and may be affected by authorization and authentication weaknesses when security controls are incomplete [OWASP Foundation 2023]. To mitigate these risks, this paper describes the deployment of a REST API in a homelab using Cloudflare Tunnel as an intermediary access layer. This approach is aligned with the principle of reducing implicit trust in network location, which is central to zero-trust architectures [Rose et al. 2020], and relies on an outbound tunnel that connects local resources to Cloudflare without requiring a publicly routable IP address [Cloudflare 2026].

This work aims to demonstrate a simple, secure, and economically viable deployment configuration for small backend applications. The case study is based on an API

developed in TypeScript with PostgreSQL as its database, hosted in a Linux virtual machine and managed through Dokku.

This work does not propose a new tunneling protocol or an alternative to the Cloudflare documentation. Its contribution is an end-to-end and replicable case study that integrates application-level controls, virtual-machine isolation, SSH hardening, Dokku-based deployment, database provisioning, and outbound-only tunnel exposure. In addition to documenting the deployment sequence, the study identifies the security, cost, latency, and availability trade-offs involved in adopting this architecture for low-demand applications.

2. Background

Representational State Transfer Application Programming Interfaces (REST APIs) handle authentication, business rules, and data access, requiring basic controls even in small applications. In this study, the API uses bearer-token authentication, Cross-Origin Resource Sharing (CORS), and rate limiting to mitigate risks such as broken authentication, authorization flaws, and unrestricted resource consumption [OWASP Foundation 2023, Tanveer et al. 2025]. These concerns are also discussed in SBSeg work on Web application security testing [Sacramento et al. 2024].

Homelabs support experimentation, learning, and low-cost deployment, but residential networks lack many protections found in cloud platforms, such as managed firewalls, Distributed Denial-of-Service (DDoS) mitigation, monitoring, redundancy, and incident response [Brush et al. 2012]. Opening router ports may expose internal services to scans, unauthorized access attempts, and configuration errors, requiring architectures that reduce public exposure.

Cloudflare Tunnel addresses this issue through a local connector that establishes an outbound encrypted tunnel to Cloudflare, which then routes public requests to the internal service [Cloudflare 2026]. This design is consistent with zero-trust principles, where access decisions should not depend only on network location [Rose et al. 2020, Syed et al. 2022, Ferretti et al. 2021]. Here, it is used as the public entry point for the homelab-hosted REST API without exposing the residential router to inbound traffic.

3. Deployment Architecture and Secure Configuration

This section describes the application, infrastructure, deployment tools, and configuration steps used in the case study. The goal is not to propose a new security mechanism, but to organize a practical and replicable deployment strategy for small backend applications hosted in residential environments. The proposed architecture combines virtualization, a lightweight Platform-as-a-Service layer, database provisioning, operating system hardening, and tunnel-based exposure to reduce the risks associated with publishing services from a home network.

3.1. Application and Infrastructure

The evaluated application is a REST API developed in TypeScript to support the AppEscolas project. The API uses PostgreSQL as its database and was originally hosted in a cloud platform. Due to cost constraints and the non-profit nature of the project, the application was migrated to a local homelab. PostgreSQL was selected because it is a

mature open-source relational database system with extensive documentation and broad support in web application deployments [PostgreSQL Global Development Group 2026].

The homelab infrastructure was already operational and based on VMware ESXi. A new virtual machine was created with 2 GB of RAM, 3 virtual CPUs from an Intel Xeon E5630 processor, and Ubuntu Server 22.04.5 as the operating system. The virtual machine was configured to host both the application and its database, while Dokku was used to simplify deployment and service management. The use of virtualization allowed the application environment to be isolated from the underlying physical host and configured with explicit CPU, memory, storage, and network parameters, following the common role of ESXi as a hypervisor for running virtual machines [Broadcom 2026].

3.2. Deployment Stack

Dokku was adopted as a lightweight Platform-as-a-Service layer. It provides a simplified deployment workflow based on Git, automatic application detection, process management, logs, and support for plugins such as PostgreSQL. In this case study, Dokku was responsible for deploying the TypeScript application, managing environment variables, and linking the application to the database service. This choice follows Dokku's deployment model, in which applications can be published through Git-based workflows and managed with platform commands [Dokku 2026a].

Ubuntu Server was selected due to its low resource consumption, compatibility with virtualized environments, and broad documentation support. PostgreSQL was provisioned through the Dokku plugin ecosystem and attached to the application after database creation. The deployment workflow was configured through Git-based synchronization, allowing the application to be updated without manual file copying to the server. The PostgreSQL plugin was used because Dokku supports service plugins that can be linked to applications, simplifying database provisioning in small self-hosted environments [Dokku 2026b].

3.3. Secure Configuration

The secure configuration followed three main steps. First, the virtual machine was created in the existing ESXi environment, with the required CPU, memory, storage, and network parameters. Figure 1 illustrates the virtual machine configuration summary used in the deployment. Second, the operating system was configured with basic hardening decisions. Network configuration was obtained through DHCP in the local network. A dedicated deployment user was created, SSH access was enabled only through certificates, password-based authentication was disabled, and direct root login through SSH was blocked. In addition, SSH remained restricted to the local network, without port forwarding or public exposure. The server operated behind IPv4 NAT and did not rely on direct IPv6 exposure. These decisions follow common SSH hardening practices, especially the use of key-based authentication and the restriction of direct administrative access [Ubuntu 2026].

Third, Dokku and Cloudflare Tunnel were configured. Dokku was installed using its standard installation script and configured to receive application requests locally. Since domain resolution, TLS, and public access were handled through Cloudflare, the Dokku configuration focused on local application routing. Cloudflare Tunnel was then configured to map the public domain to the internal service running in the homelab. This eliminated

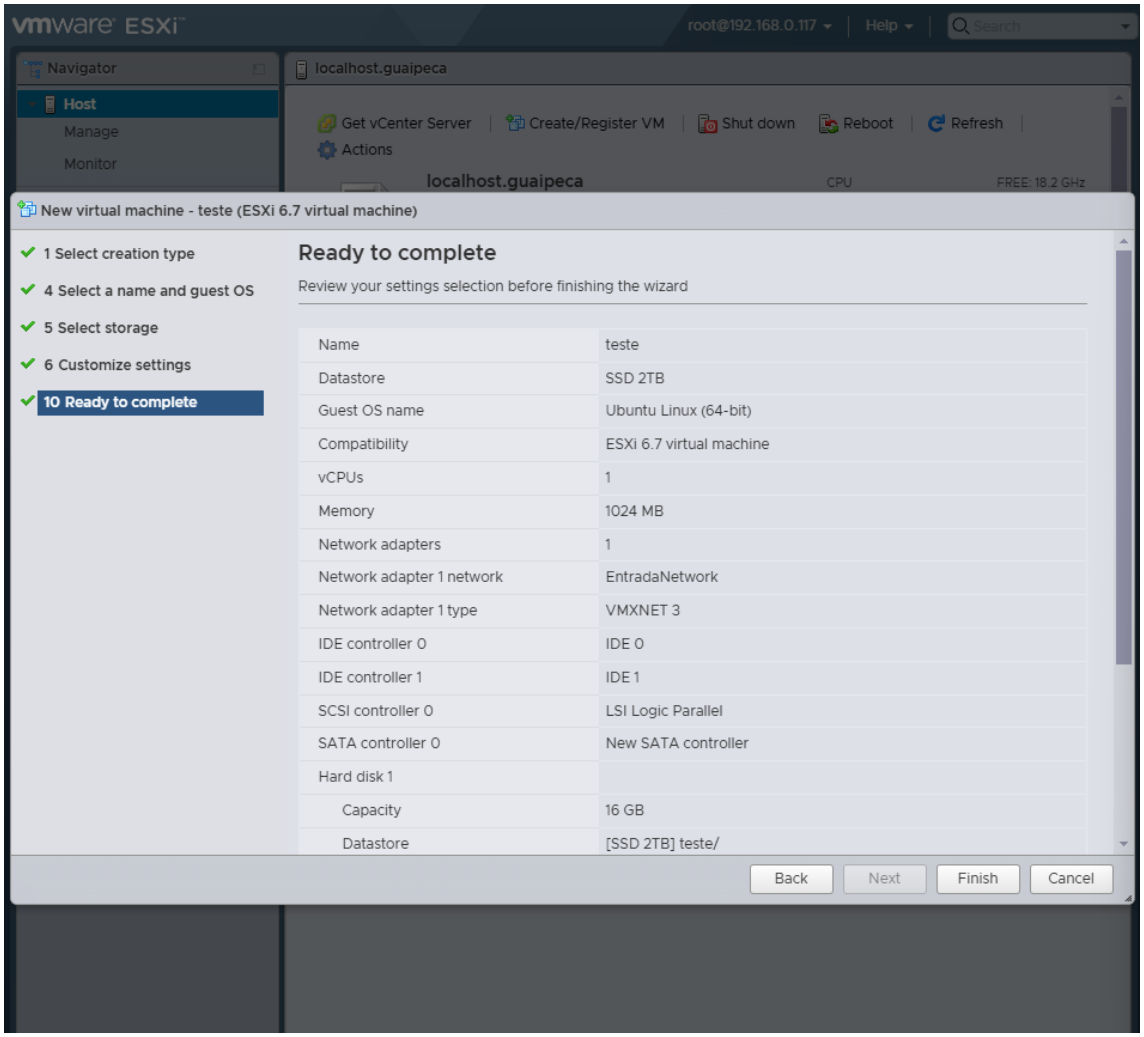

Figure 1. Virtual machine configuration summary in the homelab environment.

the need to open ports in the residential router and allowed external requests to reach the API through an outbound tunnel initiated by the local server [Cloudflare 2026].

4. Preliminary Results and Discussion

The deployment demonstrated that a small REST API can be hosted in a residential infrastructure while avoiding direct exposure of the local network. The use of Cloudflare Tunnel was the central element of the configuration, since it removed the need for router port forwarding and reduced the public attack surface of the homelab. From the developer's perspective, the resulting architecture preserved the practical benefits of a public deployment while maintaining the server behind the residential NAT.

The main positive result was the secure exposure of the API without opening inbound ports. This configuration reduces the likelihood that automated scans or direct attacks reach the residential router or the internal server. In addition, the use of Cloudflare as the public entry point simplified access monitoring and provided centralized visibility over requests to the API. This is especially relevant in small projects, where developers often lack dedicated monitoring infrastructure.

Another relevant result was the economic viability of the deployment. Since the application has low demand and does not require high-performance infrastructure, the homelab was sufficient to host the backend and database. This makes the approach suitable for course projects, proof-of-concept systems, prototypes, and non-profit applications. Although the configuration requires more initial steps than a managed cloud platform, the deployment process proved to be replicable after the operating system, Dokku, database, and tunnel were configured.

The main limitation observed was the dependence on the residential environment. The server does not have redundant power, redundant internet connectivity, or professional hardware replacement guarantees. Therefore, failures in the local machine, power outages, router issues, or internet provider instability may cause application downtime. This limitation makes the proposed architecture more appropriate for low-criticality services than for systems with strict availability requirements.

A second limitation was the additional network path introduced by the tunnel. In the evaluated scenario, the homelab was located in Uruguaiana, Rio Grande do Sul, while the closest Cloudflare infrastructure was located in Porto Alegre. This introduced approximately 40 ms of additional latency in API responses. However, this latency was not significant for the evaluated application and remained comparable to the latency previously observed when the service was hosted in a foreign datacenter.

Overall, the results suggest that homelab deployments can be a reasonable alternative for small applications when combined with adequate security practices. However, this approach should not be interpreted as equivalent to professional cloud hosting. It is a trade-off between cost, control, learning value, security, and availability.

5. Conclusion

This paper presented a practical case study on the secure exposure of a REST API hosted in a homelab environment. The proposed configuration combines secure backend practices, a Linux virtual machine, Dokku-based deployment, PostgreSQL provisioning, SSH hardening, and Cloudflare Tunnel as the public access layer. The results indicate that this approach can reduce deployment costs and support practical learning while avoiding direct exposure of the residential network to the Internet. By relying on an outbound tunnel, the application remained publicly accessible without router port forwarding and benefited from centralized traffic handling through Cloudflare. However, the approach remains limited by residential infrastructure constraints, such as hardware maintenance, power availability, internet stability, and lack of redundancy. Therefore, it is best suited for educational projects, prototypes, small applications, and proof-of-concept systems without strict availability requirements. Future work includes controlled security tests, long-term latency and availability measurements, comparison with other tunnel-based mechanisms, and integration of monitoring and alerting tools.

References

Broadcom (2026). VMware ESXi Installation and Setup Documentation. `https://techdocs.broadcom.com/us/en/vmware-cis/vsphere/vsphere/7-0/esx-installation-and-setup.html`. Accessed: 2026-06-04.

Brush, A. J. B., Jung, J., Mahajan, R., and Scott, J. (2012). Homelab: Shared infrastructure for home technology field studies. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM.

Cloudflare (2026). Cloudflare Tunnel Documentation. `https://developers.cloudflare.com/cloudflare-one/networks/connectors/cloudflare-tunnel/`. Accessed: 2026-06-04.

Dokku (2026a). Deploying an Application. `https://dokku.com/docs/deployment/application-deployment/`. Accessed: 2026-06-04.

Dokku (2026b). Dokku PostgreSQL Plugin. `https://github.com/dokku/dokku-postgres`. Accessed: 2026-06-04.

Ferretti, L., Magnanini, F., Andreolini, M., and Colajanni, M. (2021). Survivable zero trust for cloud computing environments. *Computers & Security*, 110:102419.

OWASP Foundation (2023). OWASP API Security Top 10 – 2023. `https://owasp.org/API-Security/editions/2023/en/0x11-t10/`. Accessed: 2026-06-04.

PostgreSQL Global Development Group (2026). PostgreSQL Documentation. `https://www.postgresql.org/docs/`. Accessed: 2026-06-04.

Rose, S., Borchert, O., Mitchell, S., and Connelly, S. (2020). Zero trust architecture. Technical Report NIST Special Publication 800-207, National Institute of Standards and Technology.

Sacramento, L. S. C. et al. (2024). Automação de autenticação para testes de segurança em aplicações web no zap. In *Anais do XXIV Simpósio Brasileiro de Cibersegurança*, pages 1–14. Sociedade Brasileira de Computação.

Syed, N. F., Shah, S. W., Shaghaghi, A., Anwar, A., Baig, Z., and Doss, R. (2022). Zero trust architecture (zta): A comprehensive survey. *IEEE Access*, 10:57143–57179.

Tanveer, F. et al. (2025). Towards secure apis: A survey on restful api vulnerability detection. *Computers, Materials & Continua*, 84(3).

Ubuntu (2026). OpenSSH Server Documentation. `https://documentation.ubuntu.com/server/how-to/security/openssh-server/`. Accessed: 2026-06-04.