Diário de Bordo: Construindo Meu Servidor Pessoal com Raspberry Pi

Um relato sobre aprendizado, automação e a busca pela soberania digital.

Vitor Oliveira Costa Dias | vitorcosta.site | vitorxd12.12@gmail.com
Este documento foi feito com o auxílio de ferramentas de IA.

Resumo Executivo (A Jornada em um Parágrafo)

Este documento é o diário de bordo de um projeto nascido da curiosidade: transformar um simples Raspberry Pi em um servidor web completo, seguro e automatizado. O objetivo principal sempre foi o aprendizado prático em Linux, redes, desenvolvimento e DevOps. Partindo de um hardware de baixo custo e uma instalação limpa, a jornada nos levou a containerizar uma aplicação React com Docker, servi-la com Nginx e expô-la ao mundo de uma forma surpreendentemente segura usando o Cloudflare Tunnel, tudo sem abrir uma única porta no roteador. Pelo caminho, superamos desafios de permissões, configurações de proxy e depuramos o fluxo de ponta a ponta, culminando em um pipeline de deploy automatizado. O resultado não é apenas um site hospedado em casa, mas uma plataforma robusta e um imenso campo de aprendizado para futuros projetos.

Ato I: A Visão - O Sonho de um Servidor Pessoal

Capítulo 1: A Ideia e a Justificativa

Tudo começou com uma pergunta: por que depender de plataformas de terceiros quando se pode ter controle total? O projeto nasceu do desejo de ir além dos tutoriais, de entender na prática como a internet funciona nos bastidores. A motivação era clara:

- **Aprendizado:** Mergulhar de cabeça em Linux, redes, Docker e segurança.
- **Controle:** Ter soberania sobre meus próprios dados e aplicações.
- **Customização:** Construir uma plataforma moldada exatamente para as minhas necessidades, sem as limitações de serviços gratuitos ou os custos de planos pagos.

A visão era ambiciosa, mas alcançável: um `homelab` (laboratório caseiro) que fosse ao mesmo tempo um projeto de aprendizado e uma ferramenta útil no dia a dia.

Capítulo 2: O Arsenal Inicial

Para a missão, selecionamos nosso equipamento:

- **O Hardware (O Cérebro):** Um **Raspberry Pi**. A escolha perfeita pelo baixo custo, consumo mínimo de energia e uma comunidade gigantesca. A prova de que não é preciso um data center para ter um servidor potente.
- **O Software (A Alma):** Um sistema operacional **Linux** (Raspberry Pi OS, derivado do Debian). A base sólida, aberta e infinitamente customizável que nos daria controle total sobre o ambiente.

- **A Identidade (O Endereço):** Um **domínio próprio**. Essencial para dar um nome profissional e um ponto de acesso fixo ao nosso ecossistema digital em constante evolução.

Ato II: A Execução - Poeira, Código e Conquistas

Capítulo 3: A Fortaleza Local

Com o sistema instalado, o primeiro desafio foi gerenciar um servidor "cego" (sem monitor). A solução foi o **Cockpit**, um painel de controle web que transformou nosso terminal em uma interface gráfica amigável, acessível de qualquer navegador na rede local. Organizamos o armazenamento, separando o sistema (no pendrive de boot) dos dados (no cartão SD), uma prática que garante flexibilidade e segurança. Neste ponto, tínhamos uma fortaleza local, poderosa mas isolada.

Capítulo 4: O Paradigma da Segurança

O grande dilema do `homelab`: como expor serviços para a internet sem expor a rede doméstica a ataques (IP Fixo, Port Forwarding e seus perigos). A decisão estratégica foi delegar a segurança da "fronteira" para a **Cloudflare**. Ao trocar os Nameservers, transformamos a Cloudflare de uma simples "lista telefônica" (DNS) em uma "empresa de segurança e logística" de ponta, um proxy inteligente que ficaria entre o mundo e nosso servidor.

Capítulo 5: A Passagem Secreta (Cloudflare Tunnel)

A solução para o dilema de segurança foi o **Cloudflare Tunnel**. Em vez de o mundo se conectar ao nosso Pi, nosso Pi se conecta ao mundo. O agente `cloudflared` cria um túnel criptografado persistente *de dentro para fora*, até a rede da Cloudflare. Isso inverte a lógica de conexão e é o coração da nossa segurança: o roteador doméstico permanece com todas as portas de entrada fechadas. Nenhum scanner de portas ou ataque direto pode alcançar nossa rede. É o equivalente a ter um teletransporte seguro em vez de um portão na muralha.

Capítulo 6: O Deploy Moderno (Docker e Nginx)

Para hospedar nosso site React, resistimos à tentação de instalar um servidor web diretamente no sistema. O problema a ser resolvido era o famoso "funciona na minha máquina". A solução foi o **Docker**.

1. **Dockerfile:** Criamos uma "receita" que empacota o site React (já "buildado") junto com um servidor web Nginx super leve.
2. **Contêiner:** O resultado é uma "marmita" autossuficiente e portátil, que contém o site e tudo o que ele precisa para rodar.
3. **Docker Compose:** Usamos um arquivo `docker-compose.yml` para orquestrar a execução do contêiner, definindo portas e regras de reinicialização de forma simples e declarativa.

Capítulo 7: A Automação Elegante (O Script de Deploy)

Atualizar o site envolvia uma sequência repetitiva de comandos: build, transferência, reinicialização. O problema era o trabalho manual, lento e propenso a erros. A solução foi criar um **script de deploy** (`deploy.bat`). Este "botão de deploy mágico" automatiza todo o pipeline, desde o build local até a reinicialização do contêiner no servidor, transformando o fluxo de trabalho.

Ato III: O Resultado - Um Ecossistema Vivo

Capítulo 8: A Arquitetura Final

O fluxo de uma visita ao site agora é: `Visitante` → `DNS da Cloudflare` → `Rede Segura da Cloudflare (com Redirect Rule)` → `Cloudflare Tunnel` → `Pi (agente cloudflared)` → `Contêiner Docker (Nginx)` → `Arquivos do Site React`.

Tudo isso de forma transparente, criptografada e segura.

Capítulo 9: Lições Aprendidas na Trincheira

Cada bug foi uma aula. Os principais desafios que superamos foram:

- **Lição 1: Permissões do Linux.** O clássico erro `Permission denied`. A lição: entender e usar o comando `chown` para gerenciar a propriedade de arquivos e pastas é fundamental.
- **Lição 2: Configuração de Proxy Reverso.** Os logins falhando no Cockpit. A lição: Aplicações modernas usam WebSockets e verificam cabeçalhos (`Host`, `Origin`), e o proxy precisa ser configurado para lidar com isso.
- **Lição 3: O Diabo está nos Detalhes (DNS e `config.yml`).** A frustrante página 404 do GitHub Pages ou do próprio túnel. A lição: A importância da limpeza meticulosa do DNS e da verificação de cada linha de um arquivo de configuração.
- **Lição 4: Compatibilidade de Arquitetura.** O `cloudflared` que não instalava. A lição: Um sistema `armhf` (32-bit) não é a mesma coisa que um `arm` (64-bit). Sempre verificar a arquitetura do software.

Capítulo 10: O Futuro - Ideias de Expansão

Este projeto não é um ponto final, mas uma base sólida. As próximas aventuras já estão sendo planejadas:

- **Backend e Banco de Dados:** Adicionar novos serviços ao `docker-compose.yml`, como um contêiner para uma API em Node.js ou Python e outro para um banco de dados como PostgreSQL, criando uma aplicação full-stack.
- **Alta Disponibilidade:** Implementar um Nobreak (UPS) para backup de energia e explorar um modem 4G/LTE como link de internet secundário.
- **Upgrade de Hardware:** Migrar a arquitetura para um hardware mais potente, como um mini-PC, para rodar aplicações mais pesadas (servidor de mídia, máquinas virtuais, servidor de minecraft).

- **File Server / NAS Pessoal:** Configurar um serviço de compartilhamento de arquivos (com Samba, por exemplo) para criar uma nuvem pessoal, acessível de forma segura através do túnel.
- **Projetos Físicos (IoT):** Usar o que torna o Raspberry Pi único: seus pinos GPIO. Explorar a interface I2C para conectar sensores, relés para automação residencial, ou até mesmo construir um controlador MIDI customizado.