🌀 Project Wormhole

Secure SSH + Proxy Container for Local and Remote Access

1. Conceptual Overview — How It Works

Project Wormhole provides a compact Linux container that acts as both a secure SSH endpoint and a lightweight HTTP proxy. It's designed for developers and operators who need a portable, key-secured access point for tunneling or traffic routing.

Inside the container run two services:

Component	Purpose
OpenSSH Server	Enables encrypted shell access and SSH tunnels, authenticated via your public key only (no passwords).
Tinyproxy	A minimal HTTP proxy you can use directly or through tools such as FoxyProxy in Firefox.

When you **build the image**, you inject your **public SSH key** as a build argument. Your private key remains safely on your machine. The container stores only the public key and generates unique host keys on first startup.

When the container runs, it exposes two ports:

Service	Port (in container)	Host Port (example)	Description
SSH	22	2222	Secure shell + tunneling
Proxy	8888	8888	HTTP proxy endpoint

You can then:

- Connect to it by SSH: `ssh -p 2222 paquito@localhost`
- Route browser traffic through it at `http://localhost:8888`

Optionally, you can establish **SSH tunnels** (forward or reverse) to make the proxy reachable over encrypted channels from remote machines.

2. User Guide — Using Project Wormhole

Step 1 — Build the Image

```
docker build \
  --build-arg SSH_PUBKEY="$(cat ~/.ssh/id_ed25519.pub)" \
  --build-arg USERNAME=paquito \
  -t wormhole:latest .
```

Step 2 — Run the Container

```
docker run -d --name wormhole \
  -p 2222:22 -p 8888:8888 \
  wormhole:latest
```

Check running services:

```
docker ps
```

Step 3 — Connect by SSH

Use your private key (the one matching the `.pub` you built with):

```
ssh -p 2222 paquito@localhost
```

Step 4 — Configure Firefox via FoxyProxy

1. Install **FoxyProxy Standard**.
2. In FoxyProxy → *Add New Proxy*:
   - **Title:** Wormhole
   - **Proxy Type:** HTTP
   - **Host:** `localhost`
   - **Port:** `8888`
   - (Optional) Use for all URLs
3. Enable the profile.

Test at https://ipinfo.io — you should see the container's or Docker bridge IP.

Step 5 — Optional Tunnels

Local Forward (Secure Browser Tunnel)

```
ssh -N -L 8888:localhost:8888 -p 2222 paquito@remote.example.com
```

Then keep FoxyProxy on `localhost:8888`. All traffic travels securely over SSH.

Reverse Tunnel (Remote Access to Proxy)

From inside the container:

```
ssh -N -R 0.0.0.0:9000:127.0.0.1:8888 user@remote.example.com
```

Now the proxy is reachable at `http://remote.example.com:9000`.

---

3. Appendix — Source Files

Dockerfile

```
FROM debian:bookworm-slim
ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update && apt-get install -y --no-install-recommends \
    openssh-server openssh-client tinyproxy ca-certificates dumb-init
netcat-traditional \
 && rm -rf /var/lib/apt/lists/*

ARG USERNAME=dev
ARG UID=1000
ARG GID=1000
RUN groupadd -g ${GID} ${USERNAME} \
 && useradd -m -u ${UID} -g ${GID} -s /bin/bash ${USERNAME}

ARG SSH_PUBKEY
RUN test -n "$SSH_PUBKEY" || (echo "Missing --build-arg SSH_PUBKEY"; exit
1)
RUN mkdir -p /home/${USERNAME}/.ssh \
 && echo "${SSH_PUBKEY}" > /home/${USERNAME}/.ssh/authorized_keys \
 && chmod 700 /home/${USERNAME}/.ssh \
 && chmod 600 /home/${USERNAME}/.ssh/authorized_keys \
 && chown -R ${USERNAME}:${USERNAME} /home/${USERNAME}/.ssh

RUN mkdir -p /var/run/sshd \
 && sed -ri 's/^#?PasswordAuthentication .*/PasswordAuthentication no/'
/etc/ssh/sshd_config \
 && sed -ri 's/^#?PermitRootLogin .*/PermitRootLogin no/'
/etc/ssh/sshd_config \
 && echo "AllowUsers ${USERNAME}" >> /etc/ssh/sshd_config

COPY tinyproxy.conf /etc/tinyproxy/tinyproxy.conf
COPY start.sh /usr/local/bin/start.sh
RUN chmod +x /usr/local/bin/start.sh
EXPOSE 22 8888
ENTRYPOINT ["/usr/bin/dumb-init","--"]
CMD ["/usr/local/bin/start.sh"]
```

start.sh

```
#!/usr/bin/env bash
set -euo pipefail
if [ ! -f /etc/ssh/ssh_host_rsa_key ]; then
  ssh-keygen -A
fi
chown -R "${CONTAINER_USER:-dev}:${CONTAINER_USER:-dev}"
"/home/${CONTAINER_USER:-dev}/.ssh" || true
tinyproxy -d -c /etc/tinyproxy/tinyproxy.conf &
exec /usr/sbin/sshd -D -e
```

tinyproxy.conf

```
User nobody
Group nogroup
Port 8888
Listen 0.0.0.0
LogLevel Info
Allow 0.0.0.0/0
ViaProxyName wormhole
```

Security Notes

- Only the **public key** is embedded at build time; private keys stay on your system.
- The sample proxy config allows open access — restrict the `Allow` directive in production.
- To persist logs or configs, mount `/etc/tinyproxy` or `/home/<user>` as volumes.

Summary

Project Wormhole gives you a portable, key-secured entry point for SSH access and HTTP proxying — perfect for quick tunneling, controlled remote browsing, or connecting isolated networks without exposing private services. A secure, legitimate **escape hatch** for engineers — not a backdoor.