**✅ Complete Production Setup: NGINX + systemd + Gunicorn**

Here’s how to create a fully production-ready FastAPI deployment with:

* **NGINX** as the reverse proxy.
* **systemd** to manage FastAPI as a background service.
* **Gunicorn** to handle multiple worker processes for better performance.

**🔥 1. FastAPI App: main.py**

First, make sure you have the updated FastAPI app with Docker commands (no SSH):

from fastapi import FastAPI, HTTPException

from fastapi.responses import JSONResponse

import subprocess

import json

import re

app = FastAPI()

# Docker Containers and Paths

EDGEX\_CONTAINER = "edgex-security-proxy-setup"

CHIRPSTACK\_CONTAINER = "chirpstack-chirpstack-1"

ROOT\_CONTAINER = "edgex-security-secretstore-setup"

ROOT\_FILE\_PATH = "/vault/config/assets/resp-init.json"

def run\_docker\_command(cmd, mode="generic"):

"""Run Docker command and handle output based on mode."""

try:

result = subprocess.run(

cmd, shell=True, capture\_output=True, text=True, check=True

)

output = result.stdout.strip()

# Mode-based parsing

if mode == "chirpstack":

api\_key\_data = {}

lines = output.split("\n")

for line in lines:

if line.startswith("id:"):

api\_key\_data["id"] = line.split("id:")[-1].strip()

elif line.startswith("token:"):

api\_key\_data["token"] = line.split("token:")[-1].strip()

return api\_key\_data if api\_key\_data else {"error": "Failed to parse ChirpStack API key"}

elif mode == "edgex":

try:

parsed = json.loads(output)

return {

"username": parsed.get("username", "N/A"),

"password": parsed.get("password", "No password found")

}

except json.JSONDecodeError:

return {"error": "Unexpected EdgeX response", "raw\_output": output}

elif mode == "root":

tokens = re.findall(r'"root\_token"\s\*:\s\*"([^"]+)"', output)

return tokens or {"message": "No root tokens found."}

return {"output": output}

except subprocess.CalledProcessError as e:

return {"error": f"Command failed: {e.stderr.strip()}"}

except Exception as ex:

return {"error": str(ex)}

@app.get("/")

def home():

"""Home Endpoint"""

return {"message": "Welcome to the ChirpStack, EdgeX, and Root Token Manager!"}

# EdgeX - Generate Password

@app.get("/generate-password", summary="Generate EdgeX Password", description="Generates a password for EdgeX.")

async def generate\_password(username: str):

"""Generate EdgeX password using Docker commands"""

cmd = (

f"docker exec {EDGEX\_CONTAINER} ./secrets-config proxy adduser "

f"--user \"{username}\" --tokenTTL 60 --jwtTTL 119m --useRootToken"

)

result = run\_docker\_command(cmd, mode="edgex")

return {

"username": username,

"result": result

}

# ChirpStack - Create API Key

@app.get("/create-chirpstack-api-key", summary="Create ChirpStack API Key", description="Creates an API key in ChirpStack.")

async def create\_api\_key(name: str):

"""Create ChirpStack API Key using Docker commands"""

cmd = (

f"docker exec {CHIRPSTACK\_CONTAINER} "

f"chirpstack --config /etc/chirpstack "

f"create-api-key --name '{name}'"

)

result = run\_docker\_command(cmd, mode="chirpstack")

return {

"name": name,

"result": result

}

# Root Tokens - Extract and Display

@app.get("/tokens", summary="Get Root Tokens", description="Extracts root tokens and returns them as JSON.")

def get\_tokens():

"""Extract root tokens and return them as JSON."""

cmd = f"docker exec {ROOT\_CONTAINER} cat {ROOT\_FILE\_PATH}"

tokens = run\_docker\_command(cmd, mode="root")

if isinstance(tokens, dict) and "error" in tokens:

return JSONResponse(content=tokens, status\_code=500)

return JSONResponse(content={"tokens": tokens})

**🔥 2. Gunicorn Configuration Script**

✅ Create a Gunicorn service that runs FastAPI with multiple workers.

📌 **File:** /etc/systemd/system/fastapi-daemon.service

[Unit]

Description=FastAPI App with Gunicorn

After=network.target

[Service]

User=root

WorkingDirectory=/path/to/your/fastapi/app

ExecStart=/usr/bin/gunicorn -k uvicorn.workers.UvicornWorker -w 4 -b 127.0.0.1:8000 main:app

Restart=always

[Install]

WantedBy=multi-user.target

**🔥 3. NGINX Configuration**

✅ NGINX will act as a reverse proxy to forward requests to Gunicorn.

📌 **File:** /etc/nginx/sites-available/fastapi

server {

listen 80;

server\_name 183.82.1.171;

location / {

proxy\_pass http://127.0.0.1:8000;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

# ✅ Add API docs routing

location /docs {

proxy\_pass http://127.0.0.1:8000/docs;

}

location /redoc {

proxy\_pass http://127.0.0.1:8000/redoc;

}

# Static file caching (optional)

location /static/ {

alias /path/to/your/fastapi/app/static/;

expires 30d;

}

}

# Static file caching (optional)

location /static/ {

alias /path/to/your/fastapi/app/static/;

expires 30d;

}

}

**🔥 4. Deployment Steps**

✅ **Install dependencies**:

sudo apt update

sudo apt install python3-pip python3-venv nginx gunicorn

✅ **Create and activate the virtual environment**:

cd /path/to/your/fastapi/app

python3 -m venv env

source env/bin/activate

pip install fastapi uvicorn gunicorn

✅ **Start Gunicorn with systemd**:

sudo systemctl enable fastapi-daemon

sudo systemctl start fastapi-daemon

sudo systemctl status fastapi-daemon

✅ **Enable NGINX configuration**:

sudo ln -s /etc/nginx/sites-available/fastapi /etc/nginx/sites-enabled/

sudo nginx -t

sudo systemctl restart nginx

✅ **Verify your app:**

curl http://183.82.1.171

✅ **Check logs:**

# FastAPI logs

sudo journalctl -u fastapi-daemon

# NGINX logs

sudo tail -f /var/log/nginx/access.log

sudo tail -f /var/log/nginx/error.log

**🚀 Final Architecture**

[Client] ---> [NGINX] ---> [Gunicorn with FastAPI] ---> [Docker Containers]

↑

[systemd]

**✅ Advantages of This Setup**

1. **High Performance:** Gunicorn handles multiple worker processes.
2. **Stability:** systemd ensures FastAPI automatically restarts if it crashes.
3. **Scalability:** NGINX handles SSL, static content, and can be easily scaled with load balancing.
4. **Security:** NGINX acts as a reverse proxy, protecting your FastAPI app.

🔥 Your FastAPI app is now production-ready with **NGINX, systemd, and Gunicorn**! Let me know if you need help with further optimizations, SSL setup, or scaling! 🚀