# Guia passo a passo — Deploy do Sistema de Dividendos (FastAPI + React) no AWS Lightsail (Ubuntu 22.04)

Este guia é prático, em português, e pensado para você **copiar e colar** os comandos. Ele cobre backend (FastAPI + SQLAlchemy), frontend (React + Tailwind), Nginx como proxy reverso, PostgreSQL, HTTPS (Let’s Encrypt) e automação com systemd.

**Arquitetura final**

Internet → **Nginx** (porta 80/443) → **Gunicorn/Uvicorn** (localhost:8000) → **FastAPI** → **PostgreSQL**

**Frontend**: React é “buildado” e servido como arquivos estáticos pelo Nginx em /var/www/dividendos.

## 0) Pré‑requisitos (checklist rápido)

* VM Lightsail (Ubuntu 22.04) criada e com **IP público estático** (configure no painel Lightsail).
* Acesso por **SSH** (usuário padrão ubuntu).
* (Opcional, recomendado) **Domínio** apontando para o IP público (registros A/AAAA no DNS).
* URL do seu repositório (GitHub/GitLab) **ou** arquivos do projeto prontos para upload.
* Decisão de Banco de Dados: usaremos **PostgreSQL local** neste guia.
* Tenha à mão valores para variáveis de ambiente (ex.: DATABASE\_URL, SECRET\_KEY).

**Observação**: se preferir **SQLite** temporariamente, também funciona (troque a DATABASE\_URL). Para produção, PostgreSQL é recomendado.

## 1) Conecte e faça o baseline do servidor

Conecte via SSH (pelo navegador no Lightsail ou usando seu terminal):

ssh ubuntu@SEU\_IP

Atualize o sistema e instale utilitários básicos:

sudo apt update && sudo apt -y upgrade  
sudo apt -y install git build-essential unzip curl ufw

Configurar firewall (UFW):

sudo ufw allow OpenSSH  
sudo ufw allow 'Nginx Full' # (vamos instalar o Nginx já já)  
sudo ufw enable  
sudo ufw status

**No painel do Lightsail**, certifique-se de liberar as portas **80** (HTTP) e **443** (HTTPS) na aba **Networking** da sua instância.

## 2) Instalar dependências: Python, Node.js, Nginx e PostgreSQL

### Python 3 e venv

sudo apt -y install python3 python3-venv python3-pip

### Node.js LTS (v20)

curl -fsSL https://deb.nodesource.com/setup\_20.x | sudo -E bash -  
sudo apt -y install nodejs  
node -v  
npm -v

### Nginx

sudo apt -y install nginx  
sudo systemctl enable nginx  
sudo systemctl start nginx

### PostgreSQL

sudo apt -y install postgresql postgresql-contrib  
sudo systemctl enable postgresql  
sudo systemctl start postgresql

Crie o banco e o usuário de aplicação:

sudo -u postgres psql <<'SQL'  
CREATE DATABASE dividendos;  
CREATE USER dividendos\_app WITH ENCRYPTED PASSWORD 'troque\_por\_uma\_senha\_forte';  
GRANT ALL PRIVILEGES ON DATABASE dividendos TO dividendos\_app;  
SQL

**sudo -u postgres psql**

**CREATE DATABASE dividendos;**

**CREATE USER dividendos\_app WITH ENCRYPTED PASSWORD ‘Roma@031205@p’;**

**GRANT ALL PRIVILEGES ON DATABASE dividendos TO dividendos\_app;**

**String de conexão** (exemplo):

DATABASE\_URL=postgresql+psycopg2://dividendos\_app:TROQUE@127.0.0.1:5432/dividendos

Instale o driver do PostgreSQL no Python quando formos configurar o backend (psycopg2-binary).

## 3) Estrutura de pastas sugerida

/opt/dividendos  
├─ backend/ # seu código FastAPI  
│ ├─ app/ # pacote Python (app.main, app.database, etc.)  
│ ├─ requirements.txt  
│ └─ .env # variáveis de ambiente (produção)  
├─ frontend/ # seu código React  
│ ├─ package.json  
│ └─ (src, public, ...)  
└─ releases/ # (opcional) builds antigos do frontend para rollback

Crie o diretório raiz:

sudo mkdir -p /opt/dividendos  
sudo chown -R ubuntu:www-data /opt/dividendos  
sudo chmod -R 775 /opt/dividendos

## 4) Obter o código (Git ou upload)

**Via Git** (recomendado):

cd /opt/dividendos  
git clone https://SEU\_REPO.git . # ou clone em pasta e mova

Se o seu repositório tem subpastas backend/ e frontend/, mantenha-as. Se estiver diferente, ajuste os caminhos deste guia conforme sua estrutura.

**Dica**: caso tenha desenvolvido no Windows, confira nomes de arquivos e imports **sensíveis a maiúsculas/minúsculas**. Linux é *case sensitive*.

Converta finais de linha de scripts se necessário (CRLF → LF):

sudo apt -y install dos2unix  
find /opt/dividendos -type f -name "\*.sh" -o -name "\*.py" | xargs dos2unix

## 5) Backend (FastAPI) — configurar e testar

Entre na pasta do backend, crie o virtualenv e instale dependências:

cd /opt/dividendos/backend  
python3 -m venv .venv  
source .venv/bin/activate  
pip install --upgrade pip wheel  
pip install -r requirements.txt  
# se necessário, adicione drivers (ex.):  
pip install psycopg2-binary gunicorn uvicorn

Crie o arquivo .env (produção):

cat > /opt/dividendos/backend/.env <<'ENV'  
# ===== Backend (.env produção) =====  
# Banco de Dados (PostgreSQL local)  
DATABASE\_URL=postgresql+psycopg2://dividendos\_app:TROQUE@127.0.0.1:5432/dividendos  
  
# CORS — como serviremos frontend e backend no mesmo domínio, usar \*opção segura\*  
# Se seu backend expõe /api sob o mesmo domínio, você pode restringir:  
ALLOWED\_ORIGINS=https://seu-dominio.com,https://www.seu-dominio.com,http://localhost:3000  
  
# Outros  
ENVIRONMENT=production  
LOG\_LEVEL=INFO  
SECRET\_KEY=troque\_por\_um\_valor\_bem\_aleatorio  
# Cache de preços YF (minutos) — opcional, caso implemente  
YF\_CACHE\_MINUTES=1  
ENV

**Configuração CORS no FastAPI**: garanta que seu código lê ALLOWED\_ORIGINS e configura CORSMiddleware quando necessário. Se o frontend consumir "/api" no **mesmo domínio**, CORS costuma não ser necessário.

Crie as tabelas (uma das opções abaixo):

**Com Alembic** (se seu projeto usa):

alembic upgrade head

**Sem Alembic** (exemplo simples):

python - <<'PY'  
from app.database import Base, engine  
Base.metadata.create\_all(bind=engine)  
print('Tabelas criadas (se não existiam).')  
PY

Teste localmente o backend (temporário):

uvicorn app.main:app --host 0.0.0.0 --port 8000  
# Acesse http://SEU\_IP:8000/docs (temporário; depois ficará atrás do Nginx)  
# Ctrl+C para parar

## 6) Frontend (React) — build de produção

Entre na pasta do frontend, configure a URL da API e faça o build:

Se usa **Vite**, crie .env.production com VITE\_API\_BASE\_URL=/api para usar caminho relativo (mesmo domínio):

cd /opt/dividendos/frontend  
cat > .env.production <<'ENV'  
VITE\_API\_BASE\_URL=/api  
ENV

Instale dependências e gere os arquivos estáticos:

npm ci # ou: npm install  
npm run build

Publique o build no diretório servido pelo Nginx:

sudo mkdir -p /var/www/dividendos  
sudo rsync -a --delete dist/ /var/www/dividendos/  
sudo chown -R www-data:www-data /var/www/dividendos

**Observação**: Garanta que seu frontend faz requisições para "/api" (relativo). Assim evitamos CORS e precisamos apenas de um domínio.

## 7) Nginx — site e proxy reverso

Crie a configuração do site no Nginx:

sudo tee /etc/nginx/sites-available/dividendos > /dev/null <<'NGINX'  
server {  
 listen 80;  
 listen [::]:80;  
 server\_name \_; # troque por seu domínio quando tiver (ex.: dividendos.seudominio.com)  
  
 # Frontend estático  
 root /var/www/dividendos;  
 index index.html;  
  
 # Tente servir arquivos estáticos; se não existir, entregue index.html (SPA)  
 location / {  
 try\_files $uri $uri/ /index.html;  
 }  
  
 # Proxy para o backend FastAPI em /api  
 location /api/ {  
 proxy\_pass http://127.0.0.1:8000/;  
 proxy\_http\_version 1.1;  
 proxy\_set\_header Upgrade $http\_upgrade;  
 proxy\_set\_header Connection $connection\_upgrade;  
 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;  
 client\_max\_body\_size 20m;  
 }  
  
 # Arquivos de assets estáticos (opcional para cache)  
 location ~\* \.(?:css|js|jpg|jpeg|png|gif|ico|svg|woff2?)$ {  
 expires 7d;  
 add\_header Cache-Control "public, no-transform";  
 try\_files $uri =404;  
 }  
}  
NGINX  
  
sudo ln -s /etc/nginx/sites-available/dividendos /etc/nginx/sites-enabled/dividendos  
sudo nginx -t && sudo systemctl reload nginx

Quando tiver o **domínio**, substitua server\_name \_; por server\_name SEU\_DOMINIO; e rode sudo systemctl reload nginx.

## 8) Systemd — serviço do backend (Gunicorn/Uvicorn)

Crie o serviço:

sudo tee /etc/systemd/system/dividendos-backend.service > /dev/null <<'SERVICE'  
[Unit]  
Description=Dividendos Backend (FastAPI + Gunicorn)  
After=network.target postgresql.service  
  
[Service]  
User=ubuntu  
Group=www-data  
WorkingDirectory=/opt/dividendos/backend  
EnvironmentFile=/opt/dividendos/backend/.env  
ExecStart=/opt/dividendos/backend/.venv/bin/gunicorn \  
 -k uvicorn.workers.UvicornWorker \  
 -w 2 \  
 -b 127.0.0.1:8000 \  
 app.main:app  
Restart=always  
RestartSec=5  
TimeoutStopSec=15  
  
# Segurança básica  
NoNewPrivileges=true  
PrivateTmp=true  
ProtectSystem=full  
ProtectHome=true  
  
[Install]  
WantedBy=multi-user.target  
SERVICE  
  
sudo systemctl daemon-reload  
sudo systemctl enable --now dividendos-backend  
sudo systemctl status dividendos-backend --no-pager

Se estiver tudo OK: acesse http://SEU\_IP/api/docs via Nginx.

**Dica de performance**: ajuste -w (workers) conforme vCPUs (regra aproximada: 2 x núcleos).

## 9) HTTPS (Let’s Encrypt — Certbot)

Requer domínio apontado para o IP público.

sudo apt -y install certbot python3-certbot-nginx  
sudo certbot --nginx -d SEU\_DOMINIO -d www.SEU\_DOMINIO  
# Responda o email e aceite o termo; escolha redirecionar HTTP→HTTPS  
  
# Teste o agendamento automático de renovação  
sudo certbot renew --dry-run

## 10) Importação de CSV e tarefas de manutenção

### Importar CSV manualmente (uma vez)

Coloque o arquivo CSV no servidor (use SFTP/WinSCP/scp). Exemplo:

mkdir -p /opt/dividendos/data  
# No seu computador local:  
# scp caminho/arquivo.csv ubuntu@SEU\_IP:/opt/dividendos/data/  
  
# No servidor:  
cd /opt/dividendos/backend  
source .venv/bin/activate  
python -m app.scripts.import\_csv /opt/dividendos/data/arquivo.csv

Ajuste o comando conforme seu script real de importação.

### (Opcional) Tarefa agendada com systemd timer

Se você quiser recalcular projeções **diariamente à noite**:

sudo tee /etc/systemd/system/dividendos-recalc.service > /dev/null <<'UNIT'  
[Unit]  
Description=Recalcular projeções financeiras  
After=network.target  
  
[Service]  
Type=oneshot  
User=ubuntu  
WorkingDirectory=/opt/dividendos/backend  
EnvironmentFile=/opt/dividendos/backend/.env  
ExecStart=/opt/dividendos/backend/.venv/bin/python -m app.jobs.recalc\_all  
UNIT  
  
sudo tee /etc/systemd/system/dividendos-recalc.timer > /dev/null <<'UNIT'  
[Unit]  
Description=Agendamento diário do recálculo (03:15 America/Sao\_Paulo)  
  
[Timer]  
OnCalendar=\*-\*-\* 03:15:00  
Persistent=true  
  
[Install]  
WantedBy=timers.target  
UNIT  
  
sudo systemctl daemon-reload  
sudo systemctl enable --now dividendos-recalc.timer  
systemctl list-timers | grep dividendos

Ajuste o módulo app.jobs.recalc\_all para o que existir no seu projeto (pode ser um comando/endpoint).

## 11) Atualizações futuras (deploys)

**Backend**:

cd /opt/dividendos  
git pull # ou substitua os arquivos  
cd backend  
source .venv/bin/activate  
pip install -r requirements.txt  
# migrações se houver  
# alembic upgrade head  
sudo systemctl restart dividendos-backend  
journalctl -u dividendos-backend -f

**Frontend**:

cd /opt/dividendos/frontend  
npm ci # ou npm install  
npm run build  
sudo rsync -a --delete dist/ /var/www/dividendos/

(Opcional avançado) Use /opt/dividendos/releases/timestamp/ + symlink para **rollback** rápido de versões do frontend.

## 12) Logs e troubleshooting

* Backend (systemd):

journalctl -u dividendos-backend -f

* Nginx:

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

* Testar config Nginx:

sudo nginx -t

**Problemas comuns**

* **CORS**: se o frontend chamar http://outro-dominio:8000, configure ALLOWED\_ORIGINS no backend **ou** prefira "/api" com Nginx (mesmo domínio).
* **ImportError / paths**: Linux diferencia maiúsculas/minúsculas. Confira from app.database import get\_db (nome e arquivo corretos, \_\_init\_\_.py, sem import circular).
* **Variáveis no systemd**: no EnvironmentFile, use CHAVE=valor **sem aspas**; após mudanças, sudo systemctl daemon-reload e **restart** do serviço.
* **Permissões**: garanta que ubuntu consegue ler o projeto e que /var/www/dividendos pertence ao www-data.

## 13) Hardening básico (opcional)

# Atualizações automáticas de segurança  
sudo apt -y install unattended-upgrades  
sudo dpkg-reconfigure --priority=low unattended-upgrades  
  
# Fail2ban básico  
sudo apt -y install fail2ban

Em instâncias pequenas, considere **swap** (ex.: 1–2 GB) se notar OOM. Use com cuidado em SSD (Lightsail):

sudo fallocate -l 1G /swapfile  
sudo chmod 600 /swapfile  
sudo mkswap /swapfile  
sudo swapon /swapfile  
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab

## 14) (Opcional) Docker Compose (resumo)

Se preferir empacotar tudo:

* Crie docker-compose.yml com serviços backend (Gunicorn), nginx, postgres, frontend (build e cópia para volume).
* Vantagem: portabilidade; Desvantagem: sobe a complexidade inicial.

Posso gerar um Compose pronto se você quiser.

## 15) Checklist final

* dividendos-backend.service **ativo** e servindo em 127.0.0.1:8000.
* Nginx servindo frontend em http(s)://SEU\_DOMINIO/ e proxy do backend em http(s)://SEU\_DOMINIO/api/.
* DATABASE\_URL correta e tabelas criadas.
* Importação de CSV validada (se necessário).
* Certificado HTTPS emitido e renovação ok.
* Logs sem erros (Nginx + backend).

### Próximos passos (posso ajudar!)

1. Se você me enviar **link do repositório** (ou árvore de pastas) e **módulos principais** (app.main:app, etc.), eu ajusto os arquivos de serviço e Nginx exatamente ao seu caso.
2. Posso preparar **templates prontos**: dividendos-backend.service, nginx.conf, .env e um script de deploy para você só colar.
3. Se deseja **Docker**, eu monto um docker-compose.yml completo (Postgres + backend + Nginx + build do frontend).