Portfolio Suite

Ubuntu VPS Deployment Guide

Complete step-by-step guide for deploying on Ubuntu VPS

Includes OS-level security, database multi-environment setup,

TLS configuration, and production hardening

1. Prerequisites and Server Requirements

Minimum Specifications

Resource	Development/Staging	Production
CPU	2 cores	4+ cores
RAM	4 GB	8+ GB
Storage	20 GB SSD	50+ GB SSD
Bandwidth	1 TB/month	2+ TB/month
OS	Ubuntu 22.04 LTS	Ubuntu 22.04 LTS

Domain Requirements

You need the following DNS records configured before proceeding:

Record Type	Name	Value
A	yourdomain.com	YOUR_SERVER_IP
A	www.yourdomain.com	YOUR_SERVER_IP
A	api.yourdomain.com	YOUR_SERVER_IP
A	admin.yourdomain.com	YOUR_SERVER_IP

Architecture Overview

```
                   Internet
                      |
               [Nginx Reverse Proxy]
               (TLS termination, rate limiting, security headers)
                 /        |         \
                /         |          \
     [Public Website]  [Admin UI]  [Backend API]
      React (port 3001) React (port 3000) FastAPI (port 8000)
                                       |
                            +----------+----------+
                            |          |          |
                       [PostgreSQL] [Redis]  [Celery Worker]
                        (port 5432) (port 6379)
```

2. Initial Server Access and OS-Level Security

2.1 First Login

```
# Connect to your VPS via SSH
ssh root@YOUR_SERVER_IP
```

2.2 Create a Dedicated Application User

Never run your application as root. Create a dedicated user:

```
# Create the 'portfolio' user
adduser portfolio

# Add to sudo group (for administrative tasks)
usermod -aG sudo portfolio

# Create the application directory
mkdir -p /opt/portfolio
chown portfolio:portfolio /opt/portfolio
```

2.3 Set Up SSH Key Authentication

On your local machine:

```
# Generate an SSH key pair (if you don't have one)
ssh-keygen -t ed25519 -C "your_email@example.com"

# Copy public key to the server
ssh-copy-id portfolio@YOUR_SERVER_IP
```

Test the key-based login:

```
ssh portfolio@YOUR_SERVER_IP
```

2.4 Set Timezone and Locale

```
# Set timezone to UTC (recommended for servers)
sudo timedatectl set-timezone UTC

# Verify
timedatectl

# Set locale
sudo locale-gen en_US.UTF-8
sudo update-locale LANG=en_US.UTF-8
```

2.5 Configure Automatic Security Updates

```
sudo apt install -y unattended-upgrades

# Enable automatic security updates
sudo dpkg-reconfigure -plow unattended-upgrades
# Select "Yes" when prompted

# Verify configuration
cat /etc/apt/apt.conf.d/20auto-upgrades
```

The file should contain:

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Unattended-Upgrade "1";
```

3. System Updates and Essential Packages

```
# Update package lists and upgrade existing packages
sudo apt update && sudo apt upgrade -y

# Install essential build tools and utilities
sudo apt install -y \
    build-essential \
    curl \
    wget \
    git \
    software-properties-common \
    apt-transport-https \
    ca-certificates \
    gnupg \
    lsb-release \
    htop \
    tmux \
    unzip \
    libpq-dev \
    libmagic1 \
    libffi-dev \
    libssl-dev \
    pkg-config
```

4. Firewall Configuration (UFW)

```
# Install UFW (if not already installed)
sudo apt install -y ufw

# Set default policies - deny incoming, allow outgoing
sudo ufw default deny incoming
sudo ufw default allow outgoing

# Allow SSH (IMPORTANT: do this BEFORE enabling UFW)
sudo ufw allow 22/tcp comment 'SSH'

# Allow HTTP and HTTPS
sudo ufw allow 80/tcp comment 'HTTP'
sudo ufw allow 443/tcp comment 'HTTPS'

# Enable the firewall
sudo ufw enable

# Verify the rules
sudo ufw status verbose
```

Expected output:

```
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    Anywhere       # SSH
80/tcp                     ALLOW IN    Anywhere       # HTTP
443/tcp                    ALLOW IN    Anywhere       # HTTPS
```

IMPORTANT: Do NOT open ports 5432 (PostgreSQL), 6379 (Redis), or 8000 (Uvicorn) to the public internet. These services should only be accessible from localhost.

5. SSH Hardening

Edit the SSH configuration:

```
sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.backup
sudo nano /etc/ssh/sshd_config
```

Apply these security settings:

```
# Disable root login
PermitRootLogin no

# Disable password authentication (key-only)
PasswordAuthentication no
PubkeyAuthentication yes

# Limit authentication attempts
MaxAuthTries 3

# Set idle timeout (disconnect after 10 min of inactivity)
ClientAliveInterval 300
ClientAliveCountMax 2

# Disable X11 forwarding
X11Forwarding no

# Disable TCP forwarding (unless needed)
AllowTcpForwarding no

# Restrict to specific users
AllowUsers portfolio

# Use only SSH protocol 2
Protocol 2

# Disable empty passwords
PermitEmptyPasswords no
```

Restart SSH:

```
# Test configuration first
sudo sshd -t

# If no errors, restart
sudo systemctl restart sshd
```

WARNING: Before restarting SSH, make sure you have verified that key-based login works. Otherwise you may lock

yourself out of the server.

6. Fail2Ban - Brute Force Protection

```
# Install Fail2Ban
sudo apt install -y fail2ban

# Create local configuration (never edit the main config)
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
sudo nano /etc/fail2ban/jail.local
```

Add/modify these settings:

```
[DEFAULT]
# Ban IP for 1 hour
bantime = 3600
# Detection window: 10 minutes
findtime = 600
# Max retries before ban
maxretry = 5
# Send ban notifications (optional)
# destemail = your_email@example.com
# action = %(action_mwl)s

[sshd]
enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 3
bantime = 7200

[nginx-http-auth]
enabled = true
filter = nginx-http-auth
logpath = /var/log/nginx/error.log
maxretry = 5

[nginx-limit-req]
enabled = true
filter = nginx-limit-req
logpath = /var/log/nginx/error.log
maxretry = 10
findtime = 120
bantime = 600

[nginx-botsearch]
enabled = true
filter = nginx-botsearch
logpath = /var/log/nginx/access.log
maxretry = 2
```

Start and enable Fail2Ban:

```
sudo systemctl enable fail2ban
sudo systemctl start fail2ban

# Check status
sudo fail2ban-client status
sudo fail2ban-client status sshd
```

7. Install Python 3.11+

```
# Add the deadsnakes PPA for latest Python versions
sudo add-apt-repository ppa:deadsnakes/ppa -y
sudo apt update

# Install Python 3.11 with dev packages
sudo apt install -y \
    python3.11 \
    python3.11-venv \
    python3.11-dev \
    python3.11-distutils

# Verify installation
python3.11 --version

# Install pip
curl -sS https://bootstrap.pypa.io/get-pip.py | python3.11

# Verify pip
python3.11 -m pip --version
```

8. Install Node.js 18+

```
# Install Node.js 18.x LTS using NodeSource
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install -y nodejs

# Verify installation
node --version
npm --version

# Install PM2 globally (optional - process manager for Node.js apps)
sudo npm install -g pm2
```

9. Install and Configure PostgreSQL 15+

9.1 Install PostgreSQL

```
# Add PostgreSQL APT repository
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" \
    > /etc/apt/sources.list.d/pgdg.list'

# Import repository signing key
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -

# Update and install
sudo apt update
sudo apt install -y postgresql-15 postgresql-contrib-15

# Verify installation
sudo systemctl status postgresql
psql --version
```

9.2 Secure PostgreSQL

```
# Switch to postgres user
sudo -u postgres psql
```

Inside PostgreSQL:

```
-- Set a strong password for the postgres superuser
ALTER USER postgres WITH PASSWORD 'YOUR_STRONG_POSTGRES_PASSWORD';

-- Create the application database user
CREATE USER admindb WITH PASSWORD 'YOUR_STRONG_ADMINDB_PASSWORD' CREATEDB;

-- Exit
\q
```

9.3 Configure PostgreSQL Authentication

```
# Edit pg_hba.conf
sudo nano /etc/postgresql/15/main/pg_hba.conf
```

Replace peer/trust with md5 or scram-sha-256 for local connections:

```
# TYPE  DATABASE    USER       ADDRESS        METHOD
local   all         postgres                  scram-sha-256
local   all         admindb                   scram-sha-256
host    all         admindb    127.0.0.1/32   scram-sha-256
host    all         admindb    ::1/128        scram-sha-256
```

9.4 Configure PostgreSQL for Performance

```
sudo nano /etc/postgresql/15/main/postgresql.conf
```

Adjust these settings based on your server resources:

```
# Connection Settings
listen_addresses = 'localhost'    # Only listen on localhost (SECURITY)
port = 5432
max_connections = 100

# Memory Settings (adjust based on available RAM)
shared_buffers = 2GB             # ~25% of total RAM for 8GB server
effective_cache_size = 6GB       # ~75% of total RAM
work_mem = 64MB
maintenance_work_mem = 512MB

# WAL Settings
wal_buffers = 64MB
min_wal_size = 1GB
max_wal_size = 4GB

# Checkpoint Settings
checkpoint_completion_target = 0.9

# Query Planner
random_page_cost = 1.1           # SSD storage
effective_io_concurrency = 200   # SSD storage

# Logging
log_destination = 'stderr'
logging_collector = on
log_directory = 'pg_log'
log_filename = 'postgresql-%Y-%m-%d.log'
log_rotation_age = 1d
log_rotation_size = 100MB
log_min_duration_statement = 1000  # Log queries > 1 second
log_line_prefix = '%t [%p]: [%l-1] user=%u,db=%d '

# SSL (for encrypted connections)
ssl = on
ssl_cert_file = '/etc/ssl/certs/ssl-cert-snakeoil.pem'
ssl_key_file = '/etc/ssl/private/ssl-cert-snakeoil.key'
ssl_min_protocol_version = 'TLSv1.2'
```

Restart PostgreSQL:

```
sudo systemctl restart postgresql
sudo systemctl status postgresql
```

9.5 Verify Connection

```
# Test connection with admindb user
psql -U admindb -h localhost -d postgres -W
```

10. Configure Multi-Environment Databases

The Portfolio Suite supports four environments: development, testing, staging, and production. Each uses a separate database.

10.1 Create Databases Manually (Alternative to Init Script)

```
sudo -u postgres psql
```

```
-- Create databases for each environment
CREATE DATABASE portfolioai_dev
    WITH ENCODING = 'UTF8'
    LC_COLLATE = 'C'
    LC_CTYPE = 'C'
    TEMPLATE = template0
    OWNER = admindb;

CREATE DATABASE portfolioai_test
    WITH ENCODING = 'UTF8'
    LC_COLLATE = 'C'
    LC_CTYPE = 'C'
    TEMPLATE = template0
    OWNER = admindb;

CREATE DATABASE portfolioai_staging
    WITH ENCODING = 'UTF8'
    LC_COLLATE = 'C'
    LC_CTYPE = 'C'
    TEMPLATE = template0
    OWNER = admindb;

CREATE DATABASE portfolioai
    WITH ENCODING = 'UTF8'
    LC_COLLATE = 'C'
    LC_CTYPE = 'C'
    TEMPLATE = template0
    OWNER = admindb;

-- Grant schema permissions on each database
\c portfolioai_dev
ALTER SCHEMA public OWNER TO admindb;
GRANT ALL ON SCHEMA public TO admindb;
CREATE EXTENSION IF NOT EXISTS unaccent;

\c portfolioai_test
ALTER SCHEMA public OWNER TO admindb;
GRANT ALL ON SCHEMA public TO admindb;
CREATE EXTENSION IF NOT EXISTS unaccent;

\c portfolioai_staging
ALTER SCHEMA public OWNER TO admindb;
GRANT ALL ON SCHEMA public TO admindb;
CREATE EXTENSION IF NOT EXISTS unaccent;

\c portfolioai
ALTER SCHEMA public OWNER TO admindb;
GRANT ALL ON SCHEMA public TO admindb;
CREATE EXTENSION IF NOT EXISTS unaccent;
```

```
-- Verify databases
\l

\q
```

10.2 Database URL Format

Each environment uses a connection string in this format:

```
postgresql://admindb:YOUR_PASSWORD@localhost:5432/DATABASE_NAME
```

Environment	Database Name	Connection String
Development	portfolioai_dev	`postgresql://admindb:PASSWORD@localhost:5432/p
Testing	portfolioai_test	`postgresql://admindb:PASSWORD@localhost:5432/p
Staging	portfolioai_staging	`postgresql://admindb:PASSWORD@localhost:5432/p
Production	portfolioai	`postgresql://admindb:PASSWORD@localhost:5432/p

10.3 Update alembic.ini

After deployment (see Step 15), update the database URLs in alembic.ini:

```
[development]
sqlalchemy.url = postgresql://admindb:PASSWORD@localhost:5432/portfolioai_dev

[testing]
sqlalchemy.url = postgresql://admindb:PASSWORD@localhost:5432/portfolioai_test

[staging]
sqlalchemy.url = postgresql://admindb:PASSWORD@localhost:5432/portfolioai_staging

[production]
sqlalchemy.url = postgresql://admindb:PASSWORD@localhost:5432/portfolioai
```

11. Install and Configure Redis 7

11.1 Install Redis

```
# Add Redis repository
curl -fsSL https://packages.redis.io/gpg | sudo gpg --dearmor -o /usr/share/keyrings/redis-archiv...
echo "deb [signed-by=/usr/share/keyrings/redis-archive-keyring.gpg] https://packages.redis.io/deb...
    sudo tee /etc/apt/sources.list.d/redis.list

sudo apt update
sudo apt install -y redis-server
```

11.2 Configure Redis

```
sudo nano /etc/redis/redis.conf
```

Apply these settings:

```
# Bind to localhost only (SECURITY)
bind 127.0.0.1 ::1

# Set a password
requirepass YOUR_STRONG_REDIS_PASSWORD

# Disable dangerous commands
rename-command FLUSHDB ""
rename-command FLUSHALL ""
rename-command CONFIG "CONFIG_b4d8e2f1a3c5"

# Memory management
maxmemory 256mb
maxmemory-policy allkeys-lru

# Persistence (optional but recommended)
save 900 1
save 300 10
save 60 10000
appendonly yes
appendfilename "appendonly.aof"

# Security
protected-mode yes

# Logging
loglevel notice
logfile /var/log/redis/redis-server.log
```

Restart Redis:

```
sudo systemctl restart redis-server
sudo systemctl enable redis-server

# Test connection
redis-cli -a YOUR_STRONG_REDIS_PASSWORD ping
# Should return: PONG
```

12. Install and Configure ClamAV

ClamAV provides virus scanning for file uploads. This is optional but recommended for production.

```
# Install ClamAV
sudo apt install -y clamav clamav-daemon

# Stop the daemon to update signatures
sudo systemctl stop clamav-freshclam

# Update virus definitions
sudo freshclam

# Start and enable services
sudo systemctl start clamav-freshclam
sudo systemctl enable clamav-freshclam
sudo systemctl start clamav-daemon
sudo systemctl enable clamav-daemon

# Verify ClamAV is running
sudo systemctl status clamav-daemon

# Test scanning
echo "test" > /tmp/test.txt
clamdscan /tmp/test.txt
rm /tmp/test.txt
```

13. Install and Configure Nginx

13.1 Install Nginx

```
# Install Nginx
sudo apt install -y nginx

# Enable and start
sudo systemctl enable nginx
sudo systemctl start nginx

# Verify installation
nginx -v
sudo systemctl status nginx
```

13.2 Remove Default Site

```
sudo rm /etc/nginx/sites-enabled/default
```

13.3 Harden Nginx Globally

```
sudo nano /etc/nginx/nginx.conf
```

Ensure these settings exist in the http block:

```
http {
    # Hide Nginx version
    server_tokens off;

    # Limit request body size
    client_max_body_size 10M;

    # Timeouts
    client_body_timeout 12;
    client_header_timeout 12;
    keepalive_timeout 15;
    send_timeout 10;

    # Gzip compression
    gzip on;
    gzip_vary on;
    gzip_proxied any;
    gzip_comp_level 6;
    gzip_types text/plain text/css application/json application/javascript
               text/xml application/xml application/xml+rss text/javascript
               image/svg+xml;

    # Rate limiting zones
    limit_req_zone $binary_remote_addr zone=general:10m rate=10r/s;
    limit_req_zone $binary_remote_addr zone=api:10m rate=30r/s;
    limit_req_zone $binary_remote_addr zone=auth:10m rate=5r/s;
    limit_conn_zone $binary_remote_addr zone=addr:10m;

    # Include site configurations
    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}
```

14. Obtain SSL/TLS Certificates

14.1 Install Certbot

```
sudo apt install -y certbot python3-certbot-nginx
```

14.2 Create a Temporary Nginx Config for Certificate Issuance

```
sudo nano /etc/nginx/sites-available/temp-certbot
```

```
server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com api.yourdomain.com admin.yourdomain.com;

    location /.well-known/acme-challenge/ {
        root /var/www/certbot;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}
```

```
# Create the ACME challenge directory
sudo mkdir -p /var/www/certbot

# Enable the temp site
sudo ln -s /etc/nginx/sites-available/temp-certbot /etc/nginx/sites-enabled/
sudo nginx -t && sudo systemctl reload nginx
```

14.3 Obtain Certificates

```
# Obtain certificates for all domains
sudo certbot certonly --nginx \
    -d yourdomain.com \
    -d www.yourdomain.com \
    --email your_email@example.com \
    --agree-tos \
    --no-eff-email

sudo certbot certonly --nginx \
    -d api.yourdomain.com \
    --email your_email@example.com \
    --agree-tos \
    --no-eff-email

sudo certbot certonly --nginx \
    -d admin.yourdomain.com \
    --email your_email@example.com \
    --agree-tos \
    --no-eff-email
```

14.4 Generate DH Parameters (Stronger Key Exchange)

```
sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
```

14.5 Set Up Auto-Renewal

```
# Test auto-renewal
sudo certbot renew --dry-run

# Certbot installs a systemd timer automatically
sudo systemctl status certbot.timer
```

14.6 Remove the Temporary Config

```
sudo rm /etc/nginx/sites-enabled/temp-certbot
```

15. Deploy the Application

15.1 Clone the Repository

```
# Switch to the portfolio user
sudo su - portfolio

# Clone the repository
cd /opt/portfolio
git clone https://github.com/YOUR_USERNAME/portfolio-suite.git .

# Or if using SSH
git clone git@github.com:YOUR_USERNAME/portfolio-suite.git .
```

15.2 Directory Structure on Server

```
/opt/portfolio/
|-- portfolio-backend/    # FastAPI backend
|-- backend-ui/           # React admin interface
|-- website/              # React public website
|-- deployment/           # Nginx and Docker configs
|-- maindocs/             # Documentation
|-- backups/              # Database backups (create this)
+-- logs/                 # Application logs (create this)
```

```
# Create additional directories
mkdir -p /opt/portfolio/backups
mkdir -p /opt/portfolio/logs
```

16. Backend Setup

16.1 Create Python Virtual Environment

```
cd /opt/portfolio/portfolio-backend

# Create virtual environment with Python 3.11
python3.11 -m venv venv

# Activate it
source venv/bin/activate

# Upgrade pip
pip install --upgrade pip setuptools wheel

# Install dependencies
pip install -r requirements.txt
```

16.2 Create the Backend .env File

```
nano /opt/portfolio/portfolio-backend/.env
```

```
# ============================================
# Portfolio Suite - Backend Configuration
# ============================================

# --- Environment ---
ENVIRONMENT=production
DEBUG=False
LOG_LEVEL=WARNING
LOG_FORMAT=json

# --- Database ---
DATABASE_URL=postgresql://admindb:YOUR_ADMINDB_PASSWORD@localhost:5432/portfolioai
DATABASE_URL_DEVELOPMENT=postgresql://admindb:YOUR_ADMINDB_PASSWORD@localhost:5432/portfolioai_dev
DATABASE_URL_TESTING=postgresql://admindb:YOUR_ADMINDB_PASSWORD@localhost:5432/portfolioai_test
DATABASE_URL_STAGING=postgresql://admindb:YOUR_ADMINDB_PASSWORD@localhost:5432/portfolioai_staging

# Database Pool
DB_POOL_SIZE=20
DB_MAX_OVERFLOW=0
DB_POOL_TIMEOUT=30
DB_SSL_ENABLED=True
DB_SSL_MODE=require

# --- Security ---
SECRET_KEY=GENERATE_WITH_openssl_rand_hex_32
ALGORITHM=RS256
JWT_PRIVATE_KEY_PATH=/opt/portfolio/portfolio-backend/keys/private_key.pem
JWT_PUBLIC_KEY_PATH=/opt/portfolio/portfolio-backend/keys/public_key.pem
ACCESS_TOKEN_EXPIRE_MINUTES=30
REFRESH_TOKEN_EXPIRE_MINUTES=10080

# --- CORS ---
FRONTEND_ORIGINS=https://yourdomain.com,https://www.yourdomain.com,https://admin.yourdomain.com
ALLOWED_HOSTS=api.yourdomain.com

# --- Redis ---
REDIS_URL=redis://:YOUR_REDIS_PASSWORD@localhost:6379/0

# --- Rate Limiting ---
RATE_LIMIT_ENABLED=True
RATE_LIMIT_PER_MINUTE=60
RATE_LIMIT_PER_HOUR=1000
RATE_LIMIT_PER_DAY=10000

# --- Security Headers ---
HSTS_ENABLED=True
HSTS_MAX_AGE=31536000
CSP_ENABLED=True

# --- Auto-blocking ---
```

```
AUTO_BLOCK_ENABLED=True
BLOCK_THRESHOLD_VIOLATIONS=10
BLOCK_DURATION=3600

# --- Celery ---
CELERY_BROKER_URL=redis://:YOUR_REDIS_PASSWORD@localhost:6379/0
CELERY_RESULT_BACKEND=redis://:YOUR_REDIS_PASSWORD@localhost:6379/1

# --- ClamAV ---
CLAMAV_ENABLED=true

# --- SMTP (Email) ---
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your_email@example.com
SMTP_PASSWORD=your_app_password
SMTP_TLS=True
SMTP_FROM_EMAIL=noreply@yourdomain.com
SMTP_FROM_NAME=Portfolio Suite

# --- File Uploads ---
MAX_UPLOAD_SIZE=10485760
ALLOWED_EXTENSIONS=jpg,jpeg,png,gif,pdf,doc,docx

# --- Field-Level Encryption ---
# Generate with: python -c "from cryptography.fernet import Fernet; print(Fernet.generate_key().d...
ENCRYPTION_MASTER_KEY=YOUR_FERNET_KEY
ENCRYPTION_SALT=YOUR_RANDOM_32_CHAR_SALT

# --- MFA ---
MFA_ISSUER=Portfolio Suite

# --- RAG/AI (Optional) ---
# EMBED_PROVIDER=openai
# EMBED_MODEL=text-embedding-3-small
# OPENAI_API_KEY=sk-...
```

16.3 Set Permissions on .env

```
chmod 600 /opt/portfolio/portfolio-backend/.env
```

16.4 Generate the SECRET_KEY

```
# Generate a secure secret key
openssl rand -hex 32
# Copy the output into .env as SECRET_KEY
```

16.5 Generate the Encryption Key

```
source /opt/portfolio/portfolio-backend/venv/bin/activate
python -c "from cryptography.fernet import Fernet; print(Fernet.generate_key().decode())"
# Copy the output into .env as ENCRYPTION_MASTER_KEY
```

17. Frontend Setup (Admin UI and Public Website)

17.1 Admin UI

```
cd /opt/portfolio/backend-ui

# Install dependencies
npm install

# Create .env file
nano .env
```

```
REACT_APP_API_URL=https://api.yourdomain.com
REACT_APP_API_TIMEOUT=30000
REACT_APP_IDLE_TIMEOUT_MINUTES=30
REACT_APP_TOKEN_REFRESH_INTERVAL=25
REACT_APP_MFA_ENABLED=true
REACT_APP_SECURITY_DASHBOARD_ENABLED=true
REACT_APP_FILE_UPLOADS_ENABLED=true
REACT_APP_MAX_UPLOAD_SIZE=10485760
REACT_APP_DEBUG=false
REACT_APP_LOG_LEVEL=info
FRONTEND_PORT=3000
```

```
# Build for production
npm run build
```

17.2 Public Website

```
cd /opt/portfolio/website

# Install dependencies
npm install

# Create .env file
nano .env
```

```
REACT_APP_API_URL=https://api.yourdomain.com
PORT=3001
```

```
# Build for production
npm run build
```

18. Create Systemd Services

18.1 Backend API Service

```
sudo nano /etc/systemd/system/portfolio-api.service
```

```
[Unit]
Description=Portfolio Suite - Backend API
After=network.target postgresql.service redis-server.service
Wants=postgresql.service redis-server.service

[Service]
Type=simple
User=portfolio
Group=portfolio
WorkingDirectory=/opt/portfolio/portfolio-backend
Environment="PATH=/opt/portfolio/portfolio-backend/venv/bin:/usr/local/bin:/usr/bin:/bin"
EnvironmentFile=/opt/portfolio/portfolio-backend/.env
ExecStart=/opt/portfolio/portfolio-backend/venv/bin/uvicorn \
    app.main:app \
    --host 127.0.0.1 \
    --port 8000 \
    --workers 4 \
    --access-log \
    --log-level warning
Restart=always
RestartSec=10
StandardOutput=journal
StandardError=journal
SyslogIdentifier=portfolio-api

# Security hardening
NoNewPrivileges=true
PrivateTmp=true
ProtectSystem=strict
ProtectHome=true
ReadWritePaths=/opt/portfolio/portfolio-backend/static /opt/portfolio/logs /tmp

# Resource limits
LimitNOFILE=65536
MemoryMax=2G

[Install]
WantedBy=multi-user.target
```

18.2 Celery Worker Service

```
sudo nano /etc/systemd/system/portfolio-celery.service
```

```
[Unit]
Description=Portfolio Suite - Celery Worker
After=network.target redis-server.service
Wants=redis-server.service

[Service]
Type=simple
User=portfolio
Group=portfolio
WorkingDirectory=/opt/portfolio/portfolio-backend
Environment="PATH=/opt/portfolio/portfolio-backend/venv/bin:/usr/local/bin:/usr/bin:/bin"
Environment="PROMETHEUS_MULTIPROC_DIR=/tmp/prom_multiproc"
EnvironmentFile=/opt/portfolio/portfolio-backend/.env
ExecStartPre=/bin/mkdir -p /tmp/prom_multiproc
ExecStart=/opt/portfolio/portfolio-backend/venv/bin/celery \
    -A app.queue.celery_app:get_celery \
    worker \
    --loglevel=WARNING \
    --concurrency=2
Restart=always
RestartSec=10
StandardOutput=journal
StandardError=journal
SyslogIdentifier=portfolio-celery

# Security hardening
NoNewPrivileges=true
PrivateTmp=true
ProtectSystem=strict
ProtectHome=true
ReadWritePaths=/opt/portfolio/portfolio-backend /tmp

# Resource limits
MemoryMax=1G
CPUQuota=50%

[Install]
WantedBy=multi-user.target
```

18.3 Admin UI Service (Serve with serve)

Install the serve package to host the built React apps:

```
sudo npm install -g serve
```

```
sudo nano /etc/systemd/system/portfolio-admin.service
```

```
[Unit]
Description=Portfolio Suite - Admin UI
After=network.target

[Service]
Type=simple
User=portfolio
Group=portfolio
WorkingDirectory=/opt/portfolio/backend-ui
ExecStart=/usr/bin/serve -s build -l 3000
Restart=always
RestartSec=10
StandardOutput=journal
StandardError=journal
SyslogIdentifier=portfolio-admin

[Install]
WantedBy=multi-user.target
```

18.4 Public Website Service

```
sudo nano /etc/systemd/system/portfolio-website.service
```

```
[Unit]
Description=Portfolio Suite - Public Website
After=network.target

[Service]
Type=simple
User=portfolio
Group=portfolio
WorkingDirectory=/opt/portfolio/website
ExecStart=/usr/bin/serve -s build -l 3001
Restart=always
RestartSec=10
StandardOutput=journal
StandardError=journal
SyslogIdentifier=portfolio-website

[Install]
WantedBy=multi-user.target
```

18.5 Enable and Start All Services

```
# Reload systemd daemon
sudo systemctl daemon-reload

# Enable all services
sudo systemctl enable portfolio-api
sudo systemctl enable portfolio-celery
sudo systemctl enable portfolio-admin
sudo systemctl enable portfolio-website

# Start all services
sudo systemctl start portfolio-api
sudo systemctl start portfolio-celery
sudo systemctl start portfolio-admin
sudo systemctl start portfolio-website

# Check status
sudo systemctl status portfolio-api
sudo systemctl status portfolio-celery
sudo systemctl status portfolio-admin
sudo systemctl status portfolio-website
```

19. Configure Nginx Virtual Hosts

19.1 Backend API Virtual Host

```
sudo nano /etc/nginx/sites-available/portfolio-api
```

```
upstream backend_api {
    server 127.0.0.1:8000;
    keepalive 32;
}

# HTTP to HTTPS redirect
server {
    listen 80;
    listen [::]:80;
    server_name api.yourdomain.com;

    location /.well-known/acme-challenge/ {
        root /var/www/certbot;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}

# HTTPS
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name api.yourdomain.com;

    # SSL Certificates
    ssl_certificate /etc/letsencrypt/live/api.yourdomain.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/api.yourdomain.com/privkey.pem;
    ssl_trusted_certificate /etc/letsencrypt/live/api.yourdomain.com/chain.pem;

    # TLS Configuration
    ssl_protocols TLSv1.3;
    ssl_prefer_server_ciphers off;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;
    ssl_session_tickets off;

    # OCSP Stapling
    ssl_stapling on;
    ssl_stapling_verify on;
    resolver 8.8.8.8 8.8.4.4 valid=300s;
    resolver_timeout 5s;

    # Security Headers
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains; preload" always;
    add_header X-Frame-Options "DENY" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header Referrer-Policy "strict-origin-when-cross-origin" always;
    add_header Permissions-Policy "geolocation=(), microphone=(), camera=()" always;

    server_tokens off;
    client_max_body_size 10M;
    limit_conn addr 10;

    # Auth endpoints (stricter rate limit)
    location /api/auth/ {
        limit_req zone=auth burst=10 nodelay;
        proxy_pass http://backend_api;
        proxy_http_version 1.1;
        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;
        proxy_set_header X-Request-ID $request_id;
        proxy_connect_timeout 60s;
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;
    }

    # API endpoints
    location /api/ {
        limit_req zone=api burst=20 nodelay;
        proxy_pass http://backend_api;
        proxy_http_version 1.1;
        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;
        proxy_set_header X-Request-ID $request_id;
        proxy_connect_timeout 60s;
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;
    }

    # Health checks (no rate limiting)
    location ~ ^/(healthz|readyz|metrics)$ {
        proxy_pass http://backend_api;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        access_log off;
    }

    # Static uploads
    location /uploads/ {
        alias /opt/portfolio/portfolio-backend/static/uploads/;
        expires 30d;
        add_header Cache-Control "public, immutable";
        location ~ \.php$ { deny all; }
    }
    access_log /var/log/nginx/api_access.log combined;
    error_log /var/log/nginx/api_error.log warn;
}
```

19.2 Public Website Virtual Host

```
sudo nano /etc/nginx/sites-available/portfolio-website
```

```
upstream frontend_website {
    server 127.0.0.1:3001;
    keepalive 32;
}

server {
    listen 80;
    listen [::]:80;
    server_name yourdomain.com www.yourdomain.com;

    location /.well-known/acme-challenge/ {
        root /var/www/certbot;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name yourdomain.com www.yourdomain.com;

    ssl_certificate /etc/letsencrypt/live/yourdomain.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/yourdomain.com/privkey.pem;
    ssl_trusted_certificate /etc/letsencrypt/live/yourdomain.com/chain.pem;

    ssl_protocols TLSv1.3;
    ssl_prefer_server_ciphers off;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;
    ssl_session_tickets off;
    ssl_stapling on;
    ssl_stapling_verify on;
    resolver 8.8.8.8 8.8.4.4 valid=300s;

    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains; preload" always;
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header Referrer-Policy "strict-origin-when-cross-origin" always;

    server_tokens off;
    limit_conn addr 10;

    location / {
        limit_req zone=general burst=20 nodelay;
        proxy_pass http://frontend_website;
        proxy_http_version 1.1;
```

```
            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;
        }

        location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg|woff|woff2|ttf|eot)$ {
            proxy_pass http://frontend_website;
            expires 1y;
            add_header Cache-Control "public, immutable";
        }

        access_log /var/log/nginx/website_access.log combined;
        error_log /var/log/nginx/website_error.log warn;
    }
```

19.3 Admin UI Virtual Host

```
sudo nano /etc/nginx/sites-available/portfolio-admin
```

```
upstream frontend_admin {
    server 127.0.0.1:3000;
    keepalive 32;
}

server {
    listen 80;
    listen [::]:80;
    server_name admin.yourdomain.com;

    location /.well-known/acme-challenge/ {
        root /var/www/certbot;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name admin.yourdomain.com;

    ssl_certificate /etc/letsencrypt/live/admin.yourdomain.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/admin.yourdomain.com/privkey.pem;
    ssl_trusted_certificate /etc/letsencrypt/live/admin.yourdomain.com/chain.pem;

    ssl_protocols TLSv1.3;
    ssl_prefer_server_ciphers off;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;
    ssl_session_tickets off;
    ssl_stapling on;
    ssl_stapling_verify on;
    resolver 8.8.8.8 8.8.4.4 valid=300s;

    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains; preload" always;
    add_header X-Frame-Options "DENY" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header Referrer-Policy "strict-origin-when-cross-origin" always;

    server_tokens off;
    limit_conn addr 10;

    location / {
        limit_req zone=general burst=20 nodelay;
        proxy_pass http://frontend_admin;
        proxy_http_version 1.1;
        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;
    }

    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg|woff|woff2|ttf|eot)$ {
        proxy_pass http://frontend_admin;
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    access_log /var/log/nginx/admin_access.log combined;
    error_log /var/log/nginx/admin_error.log warn;
}
```

19.4 Enable Sites and Restart Nginx

```
# Create symlinks to enable sites
sudo ln -s /etc/nginx/sites-available/portfolio-api /etc/nginx/sites-enabled/
sudo ln -s /etc/nginx/sites-available/portfolio-website /etc/nginx/sites-enabled/
sudo ln -s /etc/nginx/sites-available/portfolio-admin /etc/nginx/sites-enabled/

# Test configuration
sudo nginx -t

# If test passes, reload Nginx
sudo systemctl reload nginx
```

20. Database Initialization and Migrations

20.1 Using the Automated Init Script

The project includes an automated database initialization script that creates databases, runs migrations, and populates initial data:

```
cd /opt/portfolio/portfolio-backend
source venv/bin/activate

# Initialize all environments at once
python scripts/db/db_init.py \
    --env all \
    --username systemadmin \
    --password 'YourSecureAdminPassword!' \
    --email admin@yourdomain.com \
    --admindb-password 'YOUR_ADMINDB_PASSWORD'
```

Or initialize a specific environment:

```
# Production only
python scripts/db/db_init.py \
    --env production \
    --username systemadmin \
    --password 'YourSecureAdminPassword!' \
    --email admin@yourdomain.com
```

20.2 Manual Migration (Alternative)

```
cd /opt/portfolio/portfolio-backend
source venv/bin/activate

# Run migrations for a specific environment
alembic -x environment=production upgrade head

# Verify current migration state
alembic current

# View migration history
alembic history
```

20.3 Create Admin User

```
cd /opt/portfolio/portfolio-backend
source venv/bin/activate

python scripts/admin/create_admin.py
# Follow the interactive prompts
```

21. Generate JWT RSA Keys (RS256)

For production, use asymmetric JWT signing with RS256:

```
cd /opt/portfolio/portfolio-backend
source venv/bin/activate

# Create keys directory
mkdir -p keys

# Generate 4096-bit RSA key pair
python scripts/generate_rsa_keys.py --key-size 4096 --output-dir keys/

# Set restrictive permissions
chmod 600 keys/private_key.pem
chmod 644 keys/public_key.pem
chown portfolio:portfolio keys/*.pem
```

Verify the .env file references these keys:

```
ALGORITHM=RS256
JWT_PRIVATE_KEY_PATH=/opt/portfolio/portfolio-backend/keys/private_key.pem
JWT_PUBLIC_KEY_PATH=/opt/portfolio/portfolio-backend/keys/public_key.pem
```

22. Configure Automated Backups

22.1 Install Backup Service

```
# Copy systemd units
sudo cp /opt/portfolio/portfolio-backend/scripts/backup/portfolio-backup.service /etc/systemd/sys...
sudo cp /opt/portfolio/portfolio-backend/scripts/backup/portfolio-backup.timer /etc/systemd/system/

# Create backup directory
sudo mkdir -p /opt/portfolio/backups
sudo chown portfolio:portfolio /opt/portfolio/backups

# Enable and start the timer
sudo systemctl daemon-reload
sudo systemctl enable portfolio-backup.timer
sudo systemctl start portfolio-backup.timer

# Verify timer is active
sudo systemctl list-timers | grep portfolio
```

The timer runs the backup daily at 2:00 AM UTC.

22.2 Manual Backup

```
# Create a manual PostgreSQL backup
sudo -u portfolio pg_dump -U admindb -h localhost portfolioai | \
    gzip > /opt/portfolio/backups/portfolioai_$(date +%Y%m%d_%H%M%S).sql.gz
```

22.3 Backup Retention Cleanup (Cron)

```
# Add a cron job to delete backups older than 30 days
sudo -u portfolio crontab -e
```

Add:

```
# Delete backups older than 30 days at 3 AM
0 3 * * * find /opt/portfolio/backups -name "*.sql.gz" -mtime +30 -delete
```

23. Log Rotation

23.1 Nginx Log Rotation

Already handled by the system. Verify:

```
cat /etc/logrotate.d/nginx
```

23.2 Application Log Rotation

```
sudo nano /etc/logrotate.d/portfolio
```

```
/opt/portfolio/logs/*.log {
    daily
    missingok
    rotate 14
    compress
    delaycompress
    notifempty
    create 0640 portfolio portfolio
    sharedscripts
    postrotate
        systemctl reload portfolio-api > /dev/null 2>&1 || true
    endscript
}
```

23.3 Journald Size Limit

```
sudo nano /etc/systemd/journald.conf
```

Set:

```
[Journal]
SystemMaxUse=500M
SystemKeepFree=1G
MaxFileSec=1month
```

```
sudo systemctl restart systemd-journald
```

24. Monitoring and Health Checks

24.1 Quick Health Check Script

```
nano /opt/portfolio/healthcheck.sh
```

```
#!/bin/bash
echo "=== Portfolio Suite Health Check ==="
echo ""

# Check services
echo "--- Services ---"
for svc in portfolio-api portfolio-celery portfolio-admin portfolio-website nginx postgresql redi...
    STATUS=$(systemctl is-active $svc 2>/dev/null)
    if [ "$STATUS" = "active" ]; then
        echo "[OK]   $svc"
    else
        echo "[FAIL] $svc ($STATUS)"
    fi
done

echo ""
echo "--- Disk Usage ---"
df -h / | tail -1 | awk '{print "Used: "$3" / "$2" ("$5")"}'

echo ""
echo "--- Memory ---"
free -h | grep Mem | awk '{print "Used: "$3" / "$2}'

echo ""
echo "--- API Health ---"
HTTP_CODE=$(curl -s -o /dev/null -w "%{http_code}" http://127.0.0.1:8000/healthz 2>/dev/null)
if [ "$HTTP_CODE" = "200" ]; then
    echo "[OK]   API responding (HTTP $HTTP_CODE)"
else
    echo "[FAIL] API not responding (HTTP $HTTP_CODE)"
fi

echo ""
echo "--- Database ---"
if pg_isready -h localhost -U admindb -q; then
    echo "[OK]   PostgreSQL accepting connections"
else
    echo "[FAIL] PostgreSQL not responding"
fi

echo ""
echo "--- Redis ---"
REDIS_PING=$(redis-cli -a YOUR_REDIS_PASSWORD ping 2>/dev/null)
if [ "$REDIS_PING" = "PONG" ]; then
    echo "[OK]   Redis responding"
else
    echo "[FAIL] Redis not responding"
fi

echo ""
echo "--- SSL Certificates ---"
for DOMAIN in yourdomain.com api.yourdomain.com admin.yourdomain.com; do
    EXPIRY=$(echo | openssl s_client -servername $DOMAIN -connect $DOMAIN:443 2>/dev/null | \
             openssl x509 -noout -enddate 2>/dev/null | cut -d= -f2)
    if [ -n "$EXPIRY" ]; then
        echo "[OK]   $DOMAIN expires: $EXPIRY"
    else
        echo "[WARN] $DOMAIN certificate check failed"
    fi
done
```

```
chmod +x /opt/portfolio/healthcheck.sh
```

24.2 View Service Logs

```
# Backend API logs
sudo journalctl -u portfolio-api -f

# Celery worker logs
sudo journalctl -u portfolio-celery -f

# Nginx access logs
sudo tail -f /var/log/nginx/api_access.log

# PostgreSQL logs
sudo tail -f /var/log/postgresql/postgresql-15-main.log
```

25. Post-Deployment Verification Checklist

Run through this checklist after deployment:

OS-Level Security

- [ ] Non-root application user created (portfolio)
- [ ] SSH key-only authentication enabled
- [ ] SSH root login disabled
- [ ] UFW firewall enabled (only ports 22, 80, 443 open)
- [ ] Fail2Ban installed and configured
- [ ] Automatic security updates enabled
- [ ] Timezone set to UTC

Application Security

- [ ] ENVIRONMENT=production in .env
- [ ] DEBUG=False in .env
- [ ] Strong SECRET_KEY generated (64+ hex chars)
- [ ] RS256 JWT keys generated (4096-bit)
- [ ] Private key permissions set to 600
- [ ] .env file permissions set to 600
- [ ] CORS origins restricted to actual domains
- [ ] ALLOWED_HOSTS set to specific domains (not *)
- [ ] Rate limiting enabled
- [ ] HSTS enabled
- [ ] Redis password configured
- [ ] PostgreSQL using scram-sha-256 authentication

Infrastructure

- [ ] PostgreSQL listening on localhost only
- [ ] Redis listening on localhost only
- [ ] Nginx TLS 1.3 configured
- [ ] SSL certificates valid and auto-renewing
- [ ] All systemd services running
- [ ] Database backups configured

- [ ] Log rotation configured

Functional Tests

- [ ] https://api.yourdomain.com/healthz returns 200
- [ ] https://yourdomain.com loads the public website
- [ ] https://admin.yourdomain.com loads the admin UI
- [ ] User login works
- [ ] MFA enrollment works
- [ ] File upload works
- [ ] SSL Labs test gives A+ rating

26. Troubleshooting

Backend won't start

```
# Check logs
sudo journalctl -u portfolio-api -n 50 --no-pager

# Test manually
cd /opt/portfolio/portfolio-backend
source venv/bin/activate
python -c "from app.main import app; print('Import OK')"

# Check if port is in use
sudo lsof -i :8000
```

Database connection errors

```
# Check PostgreSQL is running
sudo systemctl status postgresql

# Test connection
psql -U admindb -h localhost -d portfolioai -W

# Check pg_hba.conf for auth method
sudo cat /etc/postgresql/15/main/pg_hba.conf | grep -v "^#" | grep -v "^$"
```

Nginx returns 502 Bad Gateway

```
# Backend is not running - check the service
sudo systemctl status portfolio-api

# Check Nginx error log
sudo tail -20 /var/log/nginx/api_error.log

# Test backend directly
curl -s http://127.0.0.1:8000/healthz
```

Redis connection refused

```
# Check Redis is running
sudo systemctl status redis-server

# Check Redis is listening
sudo ss -tlnp | grep 6379

# Test with password
redis-cli -a YOUR_REDIS_PASSWORD ping
```

Certificate renewal fails

```
# Test renewal
sudo certbot renew --dry-run

# Check Certbot timer
sudo systemctl status certbot.timer

# Manual renewal
sudo certbot renew
```

Migration errors

```
cd /opt/portfolio/portfolio-backend
source venv/bin/activate

# Check current migration state
alembic current

# Check for pending migrations
alembic history

# Rollback last migration
alembic downgrade -1

# Re-apply
alembic upgrade head
```

Permission denied on database operations

```
# Connect as postgres superuser and fix permissions
sudo -u postgres psql

-- Check permissions
\l
\du

-- Re-grant permissions
GRANT ALL PRIVILEGES ON DATABASE portfolioai TO admindb;
\c portfolioai
GRANT ALL ON SCHEMA public TO admindb;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO admindb;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO admindb;
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT ALL PRIVILEGES ON TABLES TO admindb;
\q
```

Service restart commands

```
# Restart individual services
sudo systemctl restart portfolio-api
sudo systemctl restart portfolio-celery
sudo systemctl restart portfolio-admin
sudo systemctl restart portfolio-website
sudo systemctl restart nginx
sudo systemctl restart postgresql
sudo systemctl restart redis-server

# Restart everything
sudo systemctl restart portfolio-api portfolio-celery portfolio-admin portfolio-website nginx
```

Quick Reference Card

Component	Port	Service Name	Config File
Backend API	8000	portfolio-api	/opt/portfolio/portfolio-backend/.env
Admin UI	3000	portfolio-admin	/opt/portfolio/backend-ui/.env
Public Website	3001	portfolio-website	/opt/portfolio/website/.env
Celery Worker	-	portfolio-celery	(shares backend .env)
PostgreSQL	5432	postgresql	/etc/postgresql/15/main/
Redis	6379	redis-server	/etc/redis/redis.conf
Nginx	80/443	nginx	/etc/nginx/sites-available/

Task	Command
View API logs	\`sudo journalctl -u portfolio-api -f\`
Restart API	\`sudo systemctl restart portfolio-api\`
Run migrations	\`cd /opt/portfolio/portfolio-backend && source venv/bin/activate && alembic up
Create backup	\`pg_dump -U admindb -h localhost portfolioai \
Check SSL expiry	\`sudo certbot certificates\`
Renew SSL	\`sudo certbot renew\`
Check all services	\`bash /opt/portfolio/healthcheck.sh\`
Firewall status	\`sudo ufw status\`
Fail2Ban banned IPs	\`sudo fail2ban-client status sshd\`

Document generated for Portfolio Suite deployment. Keep this document secure - it contains infrastructure architecture details.