FastAPI Sections 11-14: Migration & Deployment

Section 11: Database Migration with Alembic

1. What is Database Migration?

- **Purpose**: Version control for database schema
- **Benefits**:
  - Track database changes over time
  - Apply/rollback changes safely
  - Sync database across environments
  - Collaborate with team on schema changes

2. Alembic Setup

```bash
# Install Alembic
pip install alembic

# Initialize Alembic in project
alembic init alembic
```

File Structure Created:

```
alembic/
├── versions/
├── env.py
├── script.py.mako
└── alembic.ini
```

3. Alembic Configuration

alembic.ini:

```ini
# Remove this line and use env.py instead
# sqlalchemy.url = driver://user:pass@localhost/dbname
```

alembic/env.py:

python

```python
from app.models import Base
from app.database import SQLALCHEMY_DATABASE_URL

# Add your model's MetaData object here
target_metadata = Base.metadata

def run_migrations_online():
    configuration = config.get_section(config.config_ini_section)
    configuration["sqlalchemy.url"] = SQLALCHEMY_DATABASE_URL
    # ... rest of the function
```

4. Creating First Revision

bash

```bash
# Generate migration file
alembic revision --autogenerate -m "create posts table"

# Apply migration
alembic upgrade head

# Check current version
alembic current
```

5. Migration Commands

bash

```bash
# Create new migration
alembic revision --autogenerate -m "description"

# Apply migrations
alembic upgrade head          # Latest
alembic upgrade +1            # Next version
alembic upgrade revision_id   # Specific version

# Rollback migrations
alembic downgrade -1          # Previous version
alembic downgrade revision_id # Specific version
alembic downgrade base        # Initial state

# Show migration history
alembic history
alembic show revision_id
```

6. Disable SQLAlchemy Auto-Creation

python

```python
# Remove or comment out this line in main.py
# models.Base.metadata.create_all(bind=engine)
```

Section 12: Pre-Deployment Checklist

1. What is CORS?

Cross-Origin Resource Sharing

- Browser security feature
- Prevents cross-origin requests by default
- Must be configured for frontend-backend communication

python

```python
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

# CORS Configuration
origins = [
    "http://localhost:3000",  # React dev server
    "http://localhost:8080",  # Vue dev server
    "https://yourdomain.com",  # Production frontend
]

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

2. Git Setup

bash

```bash
# Initialize git repository
git init

# Create .gitignore
echo "__pycache__/" >> .gitignore
echo "venv/" >> .gitignore
echo ".env" >> .gitignore
echo "*.pyc" >> .gitignore

# Add and commit files
git add .
git commit -m "Initial commit"

# Connect to GitHub
git remote add origin https://github.com/username/repo.git
git push -u origin main
```

3. Essential Files for Deployment

requirements.txt:

```txt
fastapi==0.104.1
uvicorn==0.24.0
psycopg2-binary==2.9.9
sqlalchemy==2.0.23
alembic==1.12.1
python-jose[cryptography]==3.3.0
passlib[bcrypt]==1.7.4
python-multipart==0.0.6
```

.env (template):

```env
DATABASE_HOSTNAME=localhost
DATABASE_PORT=5432
DATABASE_PASSWORD=password
DATABASE_NAME=fastapi
DATABASE_USERNAME=postgres
SECRET_KEY=your-secret-key-here
ALGORITHM=HS256
ACCESS_TOKEN_EXPIRE_MINUTES=30
```

Section 13: Deployment on Heroku

1. Heroku Introduction

- **Platform as a Service (PaaS)**
- Automatic scaling and management
- Git-based deployment
- Free tier available (with limitations)

2. Create Heroku App

bash

```
# Install Heroku CLI
# Login to Heroku
heroku login

# Create new app
heroku create your-app-name

# Check apps
heroku apps
```

3. Procfile

Create `Procfile` in root directory:

```
web: uvicorn app.main:app --host=0.0.0.0 --port=${PORT:-5000}
```

4. Adding Postgres Database

bash

```
# Add postgres addon
heroku addons:create heroku-postgresql:mini

# Check database info
heroku pg:info

# Get database URL
heroku config:get DATABASE_URL
```

5. Environment Variables in Heroku

bash

```
# Set environment variables
heroku config:set SECRET_KEY=your-secret-key
heroku config:set ALGORITHM=HS256
heroku config:set ACCESS_TOKEN_EXPIRE_MINUTES=30

# View all config vars
heroku config

# Database URL is automatically set by Heroku
```

6. Alembic on Heroku

bash

```
# Run migrations on Heroku
heroku run alembic upgrade head

# Check migration status
heroku run alembic current
```

7. Deploy to Heroku

bash

```
# Deploy current branch
git push heroku main

# View logs
heroku logs --tail

# Open app in browser
heroku open
```

Section 14: Deployment on Ubuntu

1. Create Ubuntu VM

Options:

- DigitalOcean Droplet
- AWS EC2 Instance

- Google Cloud VM
- Azure VM

Recommended specs:

- Ubuntu 20.04 LTS or later
- 1GB RAM minimum
- 25GB storage minimum

2. Initial Server Setup

```bash
# Update packages
sudo apt update && sudo apt upgrade -y

# Install essential packages
sudo apt install -y software-properties-common
```

3. Install Python

```bash
# Install Python 3.9+
sudo apt install python3 python3-pip python3-venv -y

# Verify installation
python3 --version
pip3 --version
```

4. Install and Setup PostgreSQL

bash

```
# Install PostgreSQL
sudo apt install postgresql postgresql-contrib -y

# Start and enable PostgreSQL
sudo systemctl start postgresql
sudo systemctl enable postgresql

# Set password for postgres user
sudo passwd postgres

# Login to PostgreSQL
sudo -u postgres psql

# Create database and user
CREATE DATABASE fastapi;
CREATE USER fastapi_user WITH PASSWORD 'your_password';
GRANT ALL PRIVILEGES ON DATABASE fastapi TO fastapi_user;
\q
```

5. PostgreSQL Configuration

bash

```
# Edit PostgreSQL config
sudo nano /etc/postgresql/12/main/postgresql.conf

# Edit pg_hba.conf
sudo nano /etc/postgresql/12/main/pg_hba.conf

# Restart PostgreSQL
sudo systemctl restart postgresql
```

6. Setup Application

bash

```
# Create user for app
sudo adduser fastapi

# Switch to app user
sudo su - fastapi

# Clone repository
git clone https://github.com/username/fastapi-app.git
cd fastapi-app

# Create virtual environment
python3 -m venv venv
source venv/bin/activate

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

7. Environment Variables

bash

```
# Create .env file
nano .env

# Add environment variables
DATABASE_HOSTNAME=localhost
DATABASE_PORT=5432
DATABASE_PASSWORD=your_password
DATABASE_NAME=fastapi
DATABASE_USERNAME=fastapi_user
SECRET_KEY=your-secret-key
ALGORITHM=HS256
ACCESS_TOKEN_EXPIRE_MINUTES=30
```

8. Run Database Migrations

bash

```
# Run Alembic migrations
alembic upgrade head

# Test application
uvicorn app.main:app --host 0.0.0.0 --port 8000
```

9. Gunicorn Setup

bash

```
# Install Gunicorn
pip install gunicorn

# Test Gunicorn
gunicorn -w 4 -k uvicorn.workers.UvicornWorker app.main:app --bind 0.0.0.0:8000
```

10. Create Systemd Service

bash

```
# Create service file
sudo nano /etc/systemd/system/fastapi.service
```

Service file content:

ini

```ini
[Unit]
Description=FastAPI app
After=network.target

[Service]
Type=notify
User=fastapi
Group=fastapi
WorkingDirectory=/home/fastapi/fastapi-app
Environment=PATH=/home/fastapi/fastapi-app/venv/bin
ExecStart=/home/fastapi/fastapi-app/venv/bin/gunicorn -w 4 -k uvicorn.workers.UvicornWorker app.main:app --bind 0
ExecReload=/bin/kill -s HUP $MAINPID
KillMode=mixed
TimeoutStopSec=5
PrivateTmp=true

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

bash

```bash
# Enable and start service
sudo systemctl daemon-reload
sudo systemctl enable fastapi
sudo systemctl start fastapi
sudo systemctl status fastapi
```

11. NGINX Setup

bash

```bash
# Install NGINX
sudo apt install nginx -y

# Create NGINX config
sudo nano /etc/nginx/sites-available/fastapi
```

NGINX configuration:

nginx

```nginx
server {
    listen 80;
    server_name your-domain.com www.your-domain.com;

    location / {
        proxy_pass http://localhost:8000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header 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;
        proxy_cache_bypass $http_upgrade;
    }
}
```

bash

```bash
# Enable site
sudo ln -s /etc/nginx/sites-available/fastapi /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

12. Domain Name Setup

- Purchase domain from registrar
- Point DNS A record to server IP
- Configure DNS settings
- Wait for propagation (up to 24 hours)

13. SSL/HTTPS with Let's Encrypt

bash

```bash
# Install Certbot
sudo apt install certbot python3-certbot-nginx -y

# Obtain SSL certificate
sudo certbot --nginx -d your-domain.com -d www.your-domain.com

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

14. Firewall Configuration

bash

```bash
# Configure UFW firewall
sudo ufw allow OpenSSH
sudo ufw allow 'Nginx Full'
sudo ufw enable

# Check firewall status
sudo ufw status
```

15. Pushing Code Changes

bash

```bash
# On local machine
git add .
git commit -m "Update feature"
git push origin main

# On server
cd /home/fastapi/fastapi-app
git pull origin main
source venv/bin/activate
pip install -r requirements.txt
alembic upgrade head
sudo systemctl restart fastapi
```

Deployment Comparison

Feature	Heroku	Ubuntu VPS
Ease of Setup	Very Easy	Moderate
Cost	Free tier, then $7+/month	$5+/month
Control	Limited	Full control
Scalability	Automatic	Manual
Maintenance	Minimal	High
Performance	Good	Excellent
Custom Domain	Easy	Manual setup
SSL	Automatic	Manual (Let's Encrypt)

Best Practices

Security

- Use environment variables for sensitive data
- Enable firewall on VPS
- Keep systems updated
- Use HTTPS in production
- Regular backups

Performance

- Use connection pooling
- Implement caching
- Monitor resource usage
- Use CDN for static assets
- Database indexing

Monitoring

- Application logs
- Database performance
- Server metrics
- Uptime monitoring
- Error tracking