Coolify Setup & FastAPI Test Deployment Guide

Overview

Set up Coolify on test server (96.126.111.186) and deploy a simple FastAPI application to validate the entire pipeline.

Phase 1: Initial Server Access & Setup

Step 1: Get Access to Test Server

bash

```
# Option A: Reset password via Linode Manager
# 1. Go to cloud.linode.com
# 2. Click on 96.126.111.186 instance
# 3. Settings → Reset Root Password

# Option B: Use Lish Console
# 1. In Linode Manager, click "Launch Lish Console"
# 2. Login as root with temporary password
```

Step 2: Initial Server Configuration

bash

```bash
# SSH into server as root
ssh root@96.126.111.186

# Update system
apt update && apt upgrade -y

# Create your user account
adduser anthony
usermod -aG sudo anthony

# Setup SSH keys
mkdir -p /home/anthony/.ssh
# Copy your public key from local machine:
cat ~/.ssh/id_rsa.pub  # Copy this output

# On server, paste the key:
echo "YOUR_PUBLIC_KEY_HERE" > /home/anthony/.ssh/authorized_keys
chmod 600 /home/anthony/.ssh/authorized_keys
chown -R anthony:anthony /home/anthony/.ssh

# Test SSH access (from another terminal)
ssh anthony@96.126.111.186
```

Phase 2: Coolify Installation

Step 3: Install Docker & Coolify

bash

```bash
# SSH as anthony
ssh anthony@96.126.111.186

# Install Docker
curl -fsSL https://get.docker.com | sudo sh
sudo usermod -aG docker anthony
sudo systemctl enable docker
sudo systemctl start docker

# Logout and login again for docker group to take effect
exit
ssh anthony@96.126.111.186

# Verify Docker
docker --version
docker ps

# Install Coolify
curl -fsSL https://cdn.coollabs.io/coolify/install.sh | bash

# Wait for Coolify to start (2-3 minutes)
echo "Waiting for Coolify to initialize..."
sleep 120

# Check Coolify status
docker ps | grep coolify
```

Step 4: Access Coolify Web Interface

bash

```bash
# Get server IP (should be 96.126.111.186)
curl -s ifconfig.me

# Open in browser: http://96.126.111.186:8000
# Complete initial setup:
# 1. Create admin account (save credentials!)
# 2. Skip server addition for now
```

Phase 3: DNS Configuration

Step 5: Set Up DNS Records

bash

```bash
# In your domain registrar (where satori-ai-tech.com is managed):
# 1. Add A record: test.satori-ai-tech.com → 96.126.111.186
# 2. Add A record: coolify.test.satori-ai-tech.com → 96.126.111.186
# 3. Add wildcard: *.test.satori-ai-tech.com → 96.126.111.186

# Verify DNS propagation (may take 5-60 minutes)
nslookup test.satori-ai-tech.com
nslookup coolify.test.satori-ai-tech.com
```

Step 6: Configure Coolify Domain

bash

```bash
# In Coolify web interface (http://96.126.111.186:8000):
# 1. Go to Settings → Configuration
# 2. Set "Instance Domain" to: coolify.test.satori-ai-tech.com
# 3. Save and restart Coolify

# Wait for SSL certificate generation
# Then access via: https://coolify.test.satori-ai-tech.com
```

Phase 4: Create Test FastAPI Application

Step 7: Create Local FastAPI Test App

bash

```
# On your local machine
cd /Users/corelogic/satori-dev
mkdir -p test-apps/fastapi-hello
cd test-apps/fastapi-hello

# Create main.py
cat > main.py << 'EOF'
from fastapi import FastAPI
from fastapi.responses import HTMLResponse
import os
import socket
from datetime import datetime

app = FastAPI(title="Satori Test API", version="1.0.0")

@app.get("/")
async def root():
    return {
        "message": "Hello from Satori Test API!",
        "environment": os.getenv("ENVIRONMENT", "unknown"),
        "hostname": socket.gethostname(),
        "timestamp": datetime.now().isoformat()
    }

@app.get("/health")
async def health():
    return {"status": "healthy", "service": "fastapi-test"}

@app.get("/info")
async def info():
    return {
        "app": "FastAPI Test",
        "version": "1.0.0",
        "environment": os.getenv("ENVIRONMENT", "test"),
        "container_id": socket.gethostname(),
        "uptime": "running"
    }

@app.get("/ui", response_class=HTMLResponse)
async def ui():
    env = os.getenv("ENVIRONMENT", "test")
    return f"""
    <!DOCTYPE html>
```

```
    <html>
    <head>
        <title>Satori Test API</title>
        <style>
            body {{ font-family: Arial, sans-serif; margin: 40px; background: #f5f5f5;
            .container {{ background: white; padding: 20px; border-radius: 8px; box-sh
            .status {{ color: #4CAF50; font-weight: bold; }}
            .env {{ background: #e3f2fd; padding: 10px; border-radius: 4px; margin: 10
        </style>
    </head>
    <body>
        <div class="container">
            <h1>🚀 Satori Test API</h1>
            <p class="status">✅ Service is running</p>
            <div class="env">Environment: <strong>{env}</strong></div>
            <div class="env">Container: <strong>{socket.gethostname()}</strong></div>
            <div class="env">Time: <strong>{datetime.now().strftime("%Y-%m-%d %H:%M:%S
            <p><a href="/docs">📚 API Documentation</a></p>
            <p><a href="/health">🏥 Health Check</a></p>
            <p><a href="/info">ℹ️ Service Info</a></p>
        </div>
    </body>
    </html>
    """

if __name__ == "__main__":
    import uvicorn
    port = int(os.getenv("PORT", 8000))
    uvicorn.run(app, host="0.0.0.0", port=port)
EOF

# Create requirements.txt
cat > requirements.txt << 'EOF'
fastapi==0.104.1
uvicorn[standard]==0.24.0
EOF

# Create Dockerfile
cat > Dockerfile << 'EOF'
FROM python:3.11-slim

WORKDIR /app

# Install dependencies
```

```
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy application
COPY main.py .

# Expose port
EXPOSE 8000

# Health check
HEALTHCHECK --interval=30s --timeout=10s --start-period=5s --retries=3 \
    CMD curl -f http://localhost:8000/health || exit 1

# Set environment variables
ENV ENVIRONMENT=test
ENV PORT=8000

# Run the application
CMD ["python", "main.py"]
EOF

# Test locally (optional)
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
python main.py  # Test at http://localhost:8000
# Ctrl+C to stop, deactivate venv
```

Step 8: Create Git Repository

bash

```bash
# Initialize git repository
git init
git add .
git commit -m "Initial FastAPI test application"

# Create branches for environments
git branch test
git branch stage
git checkout main

# Create GitHub repository (replace with your GitHub username)
# Go to github.com and create new repository: fastapi-test
# Then:
git remote add origin https://github.com/YOUR_USERNAME/fastapi-test.git
git push -u origin main
git push origin test
git push origin stage
```

Phase 5: Deploy via Coolify

Step 9: Configure Application in Coolify

bash

```bash
# In Coolify web interface (https://coolify.test.satori-ai-tech.com):

# 1. Create new Project
#    - Name: "Satori Test Apps"
#    - Description: "Test applications for Coolify validation"

# 2. Add Server (localhost)
#    - Go to Servers → Add Server
#    - Name: "Test Server"
#    - IP: localhost (or 127.0.0.1)
#    - User: anthony
#    - Private Key: paste your SSH private key
#    - Test connection

# 3. Add Application
#    - In project, click "Add Resource" → "Application"
#    - Source: Public Repository
#    - Repository URL: https://github.com/YOUR_USERNAME/fastapi-test.git
#    - Branch: test
#    - Build Pack: Dockerfile
#    - Destination: Your test server
#    - Domain: fastapi.test.satori-ai-tech.com
#    - Port: 8000

# 4. Set Environment Variables
#    - ENVIRONMENT=test
#    - PORT=8000

# 5. Deploy
#    - Click "Deploy" button
#    - Watch build logs in real-time
```

Step 10: Monitor Deployment

bash

```
# Watch deployment progress in Coolify UI
# Or check Docker containers on server:
ssh anthony@96.126.111.186
docker ps
docker logs <container_name>

# Check if application is responding:
curl http://localhost:8000/health
curl http://fastapi.test.satori-ai-tech.com/health
```

Phase 6: Test Git Promotion Workflow

Step 11: Set Up dashd.sh for FastAPI

bash

```
# On local machine, edit dashd.sh
cd /Users/corelogic/satori-dev/dash

# Add fastapi-test to APP_REPOS in dashd.sh
# Edit the APP_REPOS array:
declare -A APP_REPOS=(
    [mallon]="clients/proj-mallon"
    [kozan]="clients/proj-kozan"
    [eddie]="cms"
    [alpha-omega]="clients/alpha-omega"
    [sensei-vend]="platform/sensei-vend"
    [fastapi-test]="test-apps/fastapi-hello"
)
```

Step 12: Test Git Promotion

bash

```bash
# Make a change to test promotion
cd /Users/corelogic/satori-dev/test-apps/fastapi-hello

# Modify main.py (change version or message)
sed -i '' 's/version="1.0.0"/version="1.0.1"/' main.py
sed -i '' 's/Hello from Satori Test API!/Hello from Satori Test API v1.0.1!/' main.py

# Commit to test branch
git add .
git commit -m "Update to version 1.0.1"
git push origin test

# This should automatically trigger Coolify deployment
# Watch in Coolify UI for automatic deployment

# Verify the change
curl https://fastapi.test.satori-ai-tech.com/
# Should show version 1.0.1
```

Step 13: Test Environment Promotion

bash

```bash
# Promote from test to stage using dashd.sh
./dashd.sh -promote fastapi-test test stage

# This will:
# 1. Merge test branch into stage branch
# 2. Push stage branch to GitHub
# 3. Trigger Coolify deployment to stage environment

# Set up stage application in Coolify:
# - Clone the test application
# - Change branch to "stage"
# - Change domain to: fastapi.stage.satori-ai-tech.com
# - Deploy
```

Phase 7: Validation & Testing

Step 14: Verify Complete Pipeline

bash

```
# Test all endpoints:
curl https://fastapi.test.satori-ai-tech.com/
curl https://fastapi.test.satori-ai-tech.com/health
curl https://fastapi.test.satori-ai-tech.com/info

# Test web interface:
# Open: https://fastapi.test.satori-ai-tech.com/ui

# Test stage environment (if set up):
curl https://fastapi.stage.satori-ai-tech.com/

# Check deployment history:
./dashd.sh -history fastapi-test --env=test
```

Step 15: Document Success

bash

```
# Create validation report
cat > /tmp/coolify-validation-report.txt << EOF
Coolify Setup Validation Report
Generated: $(date)

✅ Server Setup: 96.126.111.186
✅ Coolify Installation: https://coolify.test.satori-ai-tech.com
✅ DNS Configuration: *.test.satori-ai-tech.com
✅ FastAPI Deployment: https://fastapi.test.satori-ai-tech.com
✅ Git Promotion: test → stage workflow
✅ Auto-deployment: Git push triggers deployment
✅ Health Checks: /health endpoint responding
✅ SSL Certificates: Automatic HTTPS

Ready for production Go applications!
EOF

cat /tmp/coolify-validation-report.txt
```

Next Steps After Validation

Once FastAPI test is working:

1. **Apply to Legal Agent**: Use same process for your Go legal-agent application

2. **Set up Stage Server**: Provision second Linode for staging
3. **Set up Production Server**: Provision third Linode for production
4. **Scale to Multiple Clients**: Use client provisioning workflow

Troubleshooting Common Issues

DNS Not Resolving

```bash
# Check DNS propagation
dig test.satori-ai-tech.com
nslookup fastapi.test.satori-ai-tech.com

# May take 5-60 minutes to propagate
```

Coolify Not Starting

```bash
# Check Docker
docker ps | grep coolify
docker logs coolify

# Restart if needed
docker restart coolify
```

Application Not Deploying

```bash
# Check build logs in Coolify UI
# Or check container logs:
docker ps
docker logs <container_name>

# Common issues:
# - Wrong Dockerfile path
# - Missing environment variables
# - Port conflicts
```

This step-by-step guide will validate your entire Coolify pipeline with a simple FastAPI app before moving to your production Go applications.