Deployment Guide for mcp.xplaincrypto.ai

This guide provides step-by-step instructions for deploying the Enhanced Two-Tiered Multi-Agent Orchestration System to the DigitalOcean droplet at `mcp.xplaincrypto.ai`.

Prerequisites

Server Requirements

- **Operating System**: Ubuntu 20.04 LTS or later
- **RAM**: Minimum 4GB, Recommended 8GB+
- **Storage**: Minimum 20GB, Recommended 50GB+
- **CPU**: Minimum 2 cores, Recommended 4+ cores
- **Network**: Public IP with SSH access

Software Requirements

- Docker Engine 20.10+
- Docker Compose 2.0+
- Git
- SSH access to target servers

Access Requirements

- SSH key access to `mcp.xplaincrypto.ai`
- OpenAI API key
- Domain DNS configuration (if using custom domain)

Pre-Deployment Setup

1. Server Preparation

Connect to your DigitalOcean droplet:

```
ssh root@mcp.xplaincrypto.ai
```

Update the system:

```
apt update && apt upgrade -y
```

Install required packages:

```
apt install -y git curl wget unzip
```

2. Docker Installation

Install Docker Engine:

```
# Add Docker's official GPG key
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | gpg --dearmor -o /usr/share/
keyrings/docker-archive-keyring.gpg

# Add Docker repository
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | tee /etc/apt/
sources.list.d/docker.list > /dev/null

# Install Docker
apt update
apt install -y docker-ce docker-ce-cli containerd.io

# Install Docker Compose
curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(un
ame -s)-$(uname -m)" -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose

# Start and enable Docker
systemctl start docker
systemctl enable docker

# Add user to docker group (optional)
usermod -aG docker $USER
```

Verify installation:

```
docker --version
docker-compose --version
```

3. SSH Key Setup

Generate SSH keys for deployment (if not already available):

```
ssh-keygen -t ed25519 -C "orchestrator@mcp.xplaincrypto.ai" -f /root/.ssh/orchestrat-
or_key
```

Add the public key to authorized_keys:

```
cat /root/.ssh/orchestrator_key.pub >> /root/.ssh/authorized_keys
```

Deployment Steps

1. Clone Repository

```
cd /opt
git clone https://github.com/your-org/enhanced_orchestrator_v2.git
cd enhanced_orchestrator_v2
```

2. Environment Configuration

Copy and configure environment variables:

```
cp .env.example .env
```

Edit the `.env` file with your specific configuration:

```
nano .env
```

Required Configuration:

```
# Database Configuration
POSTGRES_DB=orchestrator_db
POSTGRES_USER=postgres
POSTGRES_PASSWORD=your_secure_db_password_here

# Redis Configuration
REDIS_PASSWORD=your_secure_redis_password_here

# API Configuration
API_KEY=your_secure_api_key_here
OPENAI_API_KEY=your_openai_api_key_here

# SSH Deployment Configuration
DEPLOY_HOST=mcp.xplaincrypto.ai
DEPLOY_PORT=22
DEPLOY_USER=root
DEPLOY_KEY_PATH=/app/keys/orchestrator_key

# Service Ports
MCP_PORT=8001
PROMETHEUS_PORT=9090
GRAFANA_PORT=3000

# Monitoring
GRAFANA_PASSWORD=your_secure_grafana_password_here
```

3. SSH Key Setup for Container

Create keys directory and copy SSH keys:

```
mkdir -p keys
cp /root/.ssh/orchestrator_key keys/deploy_key
cp /root/.ssh/orchestrator_key.pub keys/deploy_key.pub
chmod 600 keys/deploy_key
chmod 644 keys/deploy_key.pub
```

4. Create Required Directories

```
mkdir -p logs vector_stores projects monitoring/grafana monitoring/prometheus
```

5. Configure Monitoring (Optional)

Create Prometheus configuration:

```
cat > monitoring/prometheus.yml << 'EOF'
global:
  scrape_interval: 15s
  evaluation_interval: 15s

rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'orchestrator'
    static_configs:
      - targets: ['mcp_bridge:8001']
    metrics_path: '/metrics'
    scrape_interval: 30s
EOF
```

6. Deploy Services

Start the core services:

```
docker-compose up -d postgres redis mcp_bridge
```

Wait for services to be healthy:

```
docker-compose ps
```

Check logs:

```
docker-compose logs -f mcp_bridge
```

7. Verify Deployment

Test the API:

```
curl -H "X-API-Key: your_secure_api_key_here" http://localhost:8001/health
```

Expected response:

```
{
  "status": "healthy",
  "timestamp": "2024-01-15T10:30:00.000Z",
  "version": "2.0.0",
  "uptime_seconds": 123.45
}
```

8. Start Monitoring (Optional)

Start monitoring services:

```
docker-compose --profile monitoring up -d
```

Access Grafana at `http://mcp.xplaincrypto.ai:3000` (admin/your_grafana_password)

Firewall Configuration

Configure UFW firewall:

```
# Enable UFW
ufw enable

# Allow SSH
ufw allow 22/tcp

# Allow HTTP/HTTPS
ufw allow 80/tcp
ufw allow 443/tcp

# Allow orchestrator services
ufw allow 8001/tcp  # MCP Bridge
ufw allow 3000/tcp  # Grafana (optional)
ufw allow 9090/tcp  # Prometheus (optional)

# Check status
ufw status
```

SSL/TLS Configuration (Recommended)

Using Let's Encrypt with Nginx

Install Nginx:

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

Create Nginx configuration:

```
cat > /etc/nginx/sites-available/orchestrator << 'EOF'
server {
    listen 80;
    server_name mcp.xplaincrypto.ai;

    location / {
        proxy_pass http://localhost:8001;
        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;
    }
}
EOF
```

Enable the site:

```
ln -s /etc/nginx/sites-available/orchestrator /etc/nginx/sites-enabled/
nginx -t
systemctl reload nginx
```

Obtain SSL certificate:

```
certbot --nginx -d mcp.xplaincrypto.ai
```

Testing the Deployment

1. Test AI Module Workflow

Create a test repository with `ai-module.yaml`:

```
curl -X POST http://mcp.xplaincrypto.ai:8001/workflow \
  -H "Content-Type: application/json" \
  -H "X-API-Key: your_secure_api_key_here" \
  -d '{
    "repository_url": "https://github.com/your-org/test-web-app.git"
  }'
```

2. Test Task Prompt Workflow

```
curl -X POST http://mcp.xplaincrypto.ai:8001/workflow \
  -H "Content-Type: application/json" \
  -H "X-API-Key: your_secure_api_key_here" \
  -d '{
    "repository_url": "https://github.com/your-org/simple-flask-app.git",
    "task_prompt": "Deploy a simple Flask web server with basic authentication"
  }'
```

3. Test SSH Command Execution

```
curl -X POST http://mcp.xplaincrypto.ai:8001/execute \
  -H "Content-Type: application/json" \
  -H "X-API-Key: your_secure_api_key_here" \
  -d '{
    "command": "docker ps",
    "security_level": "medium"
  }'
```

Monitoring and Maintenance

Log Management

View logs:

```
# All services
docker-compose logs -f

# Specific service
docker-compose logs -f mcp_bridge

# System logs
tail -f /var/log/syslog
```

Rotate logs:

```
# Configure logrotate
cat > /etc/logrotate.d/orchestrator << 'EOF'
/opt/enhanced_orchestrator_v2/logs/*.log {
    daily
    missingok
    rotate 30
    compress
    delaycompress
    notifempty
    create 644 root root
}
EOF
```

Database Maintenance

Backup database:

```
docker-compose exec postgres pg_dump -U postgres orchestrator_db > backup_$(date +
%Y%m%d).sql
```

Restore database:

```
docker-compose exec -T postgres psql -U postgres orchestrator_db < backup_20240115.sql
```

Updates and Upgrades

Update the system:

```
cd /opt/enhanced_orchestrator_v2
git pull origin main
docker-compose build --no-cache
docker-compose up -d
```

Troubleshooting

Common Issues

1. **Container won’t start**
   `bash`
   `   docker-compose logs container_name`
   `   docker-compose ps`

2. **SSH connection fails**
   ```
   bash
      # Check SSH key permissions
      ls -la keys/
      # Test SSH connection manually
      ssh -i keys/deploy_key root@mcp.xplaincrypto.ai
   ```

3. **Database connection issues**
   ```
   bash
      # Check database status
      docker-compose exec postgres pg_isready -U postgres
      # Check connection from app
      docker-compose exec mcp_bridge python -c "import psycopg2; print('DB OK')"
   ```

4. **API not responding**
   ```
   bash
      # Check if service is running
      curl http://localhost:8001/health
      # Check firewall
      ufw status
      # Check nginx (if using)
      nginx -t
      systemctl status nginx
   ```

Performance Optimization

1. **Increase Docker resources**
   ```
   bash
      # Edit Docker daemon configuration
      nano /etc/docker/daemon.json
   ```

2. **Optimize PostgreSQL**
   ```
   bash
      # Tune PostgreSQL settings in docker-compose.yml
      # Add performance-related environment variables
   ```

3. **Monitor resource usage**
   ```
   bash
      docker stats
      htop
      df -h
   ```

Security Hardening

Additional Security Measures

1. **Change default ports**
2. **Implement fail2ban**
3. **Regular security updates**
4. **Monitor access logs**
5. **Use strong passwords and keys**

Backup Strategy

1. **Database backups**: Daily automated backups
2. **Configuration backups**: Version control
3. **Log archival**: Long-term storage
4. **Disaster recovery**: Documented procedures

Support and Maintenance

Regular Maintenance Tasks

- **Weekly**: Check logs and system health
- **Monthly**: Update dependencies and security patches
- **Quarterly**: Review and rotate credentials
- **Annually**: Security audit and penetration testing

Contact Information

For support and issues:
- **Documentation**: Check this guide and architecture docs
- **Logs**: Always include relevant log files
- **Environment**: Provide system and configuration details