Phase 2: Oracle Cloud VPS Deployment Guide

Complete Step-by-Step Instructions

Deploy your SMC Trading Bot to Oracle Cloud Free Tier VPS for 24/7 operation.

Prerequisites

- Completed Phase 1 (Local Docker deployment working)
- Oracle Cloud Free Tier account
- GitHub account
- SSH client installed
- Domain name (optional)

Part 1: Oracle Cloud Setup

Step 1: Create Oracle Cloud Account

1. Go to https://www.oracle.com/cloud/free/
2. Click "Start for free"
3. Fill in registration details
4. Verify email and phone
5. Add payment method (won't be charged on free tier)

Step 2: Create VPS Instance

1. Log in to Oracle Cloud Console
2. Navigate to **Compute → Instances**
3. Click **Create Instance**

Instance Configuration:

- Name: `smc-trading-bot`
- Image: **Ubuntu 22.04 Minimal**
- Shape: **VM.Standard.E2.1.Micro** (Always Free)
- Networking: Use default VCN
- Add SSH keys: Upload your public key or generate new

4. Click **Create**

5. Wait 2-3 minutes for provisioning
6. Note down **Public IP address**

Step 3: Configure Firewall Rules

1. Go to **Networking → Virtual Cloud Networks**
2. Click your VCN → **Security Lists → Default Security List**
3. Click **Add Ingress Rules**

Add these rules:

Source CIDR	Protocol	Port Range	Description
0.0.0.0/0	TCP	22	SSH
0.0.0.0/0	TCP	80	HTTP
0.0.0.0/0	TCP	443	HTTPS
0.0.0.0/0	TCP	5000	Backend API
0.0.0.0/0	TCP	3000	Frontend

4. Click **Add Ingress Rules**

Part 2: VPS Initial Setup

Step 1: Connect to VPS

```
# Replace with your VPS IP
ssh ubuntu@YOUR_VPS_IP

# If using custom key
ssh -i ~/.ssh/your-key.pem ubuntu@YOUR_VPS_IP
```

Step 2: Update System

```
sudo apt-get update
sudo apt-get upgrade -y
sudo reboot
```

Wait 2 minutes, then reconnect.

Step 3: Install Docker

```
# Install Docker
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

# Add user to docker group
sudo usermod -aG docker $USER

# Apply group changes
newgrp docker

# Verify installation
docker --version
```

Step 4: Install Docker Compose

```
# Install Docker Compose
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$

# Make executable
sudo chmod +x /usr/local/bin/docker-compose

# Verify installation
docker-compose --version
```

Step 5: Install Git

```
sudo apt-get install -y git
git --version
```

Step 6: Configure Firewall (UFW)

```
# Enable UFW
sudo ufw allow 22/tcp
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw allow 5000/tcp
sudo ufw allow 3000/tcp
sudo ufw enable

# Check status
sudo ufw status
```

Part 3: Deploy Bot to VPS

Step 1: Create GitHub Repository

1. Go to https://github.com/new
2. Repository name: `smc-trading-bot`
3. Choose **Private** (recommended)
4. Click **Create repository**

Step 2: Push Code to GitHub

On your **local machine**:

```
cd smc-trading-bot

# Initialize git (if not already)
git init

# Add GitHub remote
git remote add origin https://github.com/YOUR_USERNAME/smc-trading-bot.git

# Add all files
git add .

# Commit
git commit -m "Initial commit - SMC Trading Bot"

# Push to GitHub
git push -u origin main
```

Step 3: Clone Repository on VPS

Back on **VPS**:

```
# Create app directory
sudo mkdir -p /opt/smc-bot
sudo chown -R $USER:$USER /opt/smc-bot
cd /opt/smc-bot

# Clone repository
git clone https://github.com/YOUR_USERNAME/smc-trading-bot.git .

# Verify files
ls -la
```

Step 4: Configure Environment

```
# Copy environment template
cp .env.example .env

# Edit environment file
nano .env
```

Add your Delta Exchange credentials:

```
ENVIRONMENT_MODE=testnet
DELTA_TESTNET_API_KEY=your_actual_key_here
DELTA_TESTNET_SECRET=your_actual_secret_here
```

Save: `Ctrl+O`, `Enter`, `Ctrl+X`

Step 5: Build and Start Bot

```
# Build Docker images
docker-compose build

# Start containers
docker-compose up -d

# Check status
docker-compose ps

# View logs
docker-compose logs -f
```

Press `Ctrl+C` to exit logs.

Part 4: Systemd Service for Auto-Start

Step 1: Create Service File

```
sudo nano /etc/systemd/system/smc-bot.service
```

Add this content:

```
[Unit]
Description=SMC Trading Bot
After=docker.service
Requires=docker.service

[Service]
Type=oneshot
RemainAfterExit=yes
```

```
WorkingDirectory=/opt/smc-bot
ExecStart=/usr/local/bin/docker-compose up -d
ExecStop=/usr/local/bin/docker-compose down
User=ubuntu
Group=docker

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

Save and exit.

Step 2: Enable and Start Service

```
# Reload systemd
sudo systemctl daemon-reload

# Enable service (auto-start on boot)
sudo systemctl enable smc-bot.service

# Start service
sudo systemctl start smc-bot.service

# Check status
sudo systemctl status smc-bot.service
```

Step 3: Test Auto-Start

```
# Reboot VPS
sudo reboot

# Wait 2 minutes, then reconnect
ssh ubuntu@YOUR_VPS_IP

# Check if bot is running
docker-compose ps
```

Part 5: GitHub Actions CI/CD

Step 1: Generate SSH Key for Deployment

On **VPS**:

```
# Generate key
ssh-keygen -t rsa -b 4096 -C "github-actions" -f ~/.ssh/deploy_key -N ""

# Display public key
cat ~/.ssh/deploy_key.pub
```

Add public key to authorized_keys:

```
cat ~/.ssh/deploy_key.pub &gt;&gt; ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
```

Step 2: Get Private Key

```
# Display private key
cat ~/.ssh/deploy_key
```

Copy the entire output (including `-----BEGIN` and `-----END` lines).

Step 3: Add Secrets to GitHub

1. Go to your GitHub repository
2. Click **Settings** → **Secrets and variables** → **Actions**
3. Click **New repository secret**

Add these secrets:

Name	Value
`ORACLE_VPS_IP`	Your VPS public IP
`ORACLE_SSH_KEY`	Private key from Step 2
`DELTA_TESTNET_API_KEY`	Your testnet API key
`DELTA_TESTNET_SECRET`	Your testnet secret

Step 4: Create Workflow File

On **local machine**:

```
mkdir -p .github/workflows
nano .github/workflows/deploy.yml
```

Add this content:

```
name: Deploy to Oracle Cloud VPS

on:
  push:
    branches:
      - main
      - develop
  workflow_dispatch:

jobs:
```

```
  deploy:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v3

    - name: Deploy to VPS
      uses: appleboy/ssh-action@master
      with:
        host: ${{ secrets.ORACLE_VPS_IP }}
        username: ubuntu
        key: ${{ secrets.ORACLE_SSH_KEY }}
        script: |
          cd /opt/smc-bot
          git pull origin main
          docker-compose down
          docker-compose build --no-cache
          docker-compose up -d
          docker-compose ps
          echo "✅ Deployment complete!"
```

Save and commit:

```
git add .github/workflows/deploy.yml
git commit -m "Add GitHub Actions deployment workflow"
git push origin main
```

Step 5: Test Automated Deployment

1. Go to GitHub repository
2. Click **Actions** tab
3. You should see the workflow running
4. Wait for completion (green checkmark)

Make a test change:

```
# Edit README
echo "## Test deployment" &gt;&gt; README.md
git add README.md
git commit -m "Test automated deployment"
git push origin main
```

Check GitHub Actions - it should automatically deploy to VPS.

Part 6: Access and Monitor

Access Dashboard

Open browser: `http://YOUR_VPS_IP:3000`

Monitor Logs

```
# Backend logs
docker-compose logs -f backend

# Frontend logs
docker-compose logs -f frontend

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

Check Bot Status

```
# Via API
curl http://YOUR_VPS_IP:5000/api/status

# Via systemd
sudo systemctl status smc-bot.service

# Docker containers
docker-compose ps
```

Part 7: Optional - Domain Setup

Step 1: Point Domain to VPS

1. Go to your domain registrar
2. Add **A Record**:
   - Host: `@` or `bot`
   - Points to: `YOUR_VPS_IP`
   - TTL: 3600

Step 2: Install Nginx as Reverse Proxy

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

Step 3: Configure Nginx

```
sudo nano /etc/nginx/sites-available/smc-bot
```

Add:

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

    location / {
        proxy_pass http://localhost:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    location /api/ {
        proxy_pass http://localhost:5000/api/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

Enable site:

```
sudo ln -s /etc/nginx/sites-available/smc-bot /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

Step 4: Add SSL Certificate

```
sudo certbot --nginx -d your-domain.com
```

Follow prompts. Now access via: `https://your-domain.com`

Troubleshooting

Bot Not Starting

```
# Check Docker status
sudo systemctl status docker

# Restart Docker
sudo systemctl restart docker

# Restart bot
cd /opt/smc-bot
```

```
docker-compose down
docker-compose up -d
```

Out of Memory

```
# Check memory
free -h

# Add swap space
sudo fallocate -l 2G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

Connection Issues

```
# Check firewall
sudo ufw status

# Check if ports are listening
sudo netstat -tulpn | grep -E '3000|5000'
```

Maintenance

Update Bot

```
cd /opt/smc-bot
git pull
docker-compose down
docker-compose build
docker-compose up -d
```

View Resource Usage

```
# CPU and RAM
htop

# Docker resources
docker stats
```

Backup Configuration

```
# Backup .env file
cp .env .env.backup

# Backup to local machine
scp ubuntu@YOUR_VPS_IP:/opt/smc-bot/.env ./env-backup
```

Next Steps

✅ Phase 2 complete - bot running 24/7 on Oracle Cloud
➡ Proceed to Phase 3: RL/AI Integration

[[71]][77][[80]][83][^86]

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20]

⁂

1. https://docs.delta.exchange
2. https://help.ovhcloud.com/csm/en-vps-deploy-website-github-actions?id=kb_article_view&sysparm_article=KB0066184
3. https://deltaexchangeindia.freshdesk.com/support/solutions/articles/80001181136-getting-started-with-delta-exchange-api-copilot
4. https://docs.docker.com/reference/compose-file/services/
5. https://www.hostinger.com/support/deploy-to-hostinger-vps-using-github-actions/
6. https://www.youtube.com/watch?v=TSzz-Fllq14
7. https://learn.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/multi-container-applications-docker-compose
8. https://blog.ando.ai/posts/github-actions-vps-deployment/
9. https://www.youtube.com/watch?v=O5i5DJUAap4
10. https://docs.docker.com/compose/how-tos/multiple-compose-files/
11. https://docs.github.com/actions/deployment/about-deployments/deploying-with-github-actions
12. https://github.com/delta-exchange/python-rest-client
13. https://www.delta.exchange/support/solutions/articles/80001153884-your-complete-api-faq-guide-everything-you-need-to-know
14. https://www.delta.exchange/algo/delta-exchange-apis
15. https://www.delta.exchange/blog/introduction-to-algorithmic-trading-with-delta-rest-client
16. https://www.profitaddaweb.com/2025/04/delta-exchange-api-in-python.html
17. https://stackoverflow.com/questions/49191304/how-to-deal-with-multiple-services-inside-a-docker-compose-yml
18. https://gist.github.com/danielwetan/4f4db933531db5dd1af2e69ec8d54d8a
19. https://www.youtube.com/watch?v=RoVeASkeMpc

20. https://matthiasnoback.nl/2018/03/defining-multiple-similar-services-with-docker-compose/