🚀 Complete Deployment Guide for Trading App

Deploy to Ubuntu VPS (67.211.219.94)

This guide will walk you through **every single step** needed to deploy your trading application. Each command is explained in simple terms so you know exactly what you're doing.

📋 What You're Deploying

- **Backend**: Go application (handles data, trading logic, API)
- **Frontend**: Next.js web application (the website users see)
- **VPS Details**: IP 67.211.219.94, Ubuntu OS, 2GB RAM, 1 CPU core

---

⚠️ Before You Start

Required on your local computer:
- The deployment package: `trading-app-deployment.tar.gz` (located at `/home/ubuntu/code_artifacts/trading-app-deployment.tar.gz`)
- SSH access to your VPS (you should be able to log in)

Already installed on your VPS:
- Go programming language ✅

What we'll install on the VPS:
- Node.js and npm (for the frontend)
- System dependencies (tesseract-ocr for document processing)

📦 STEP 1: Transfer the Application to Your VPS

1.1 Open Terminal on Your Local Computer

What this does: Opens a command-line interface to run commands.

- **Mac**: Press `Cmd + Space`, type "Terminal", press Enter
- **Windows**: Use PowerShell or Windows Terminal
- **Linux**: Press `Ctrl + Alt + T`

1.2 Navigate to the Folder with the Deployment Package

What this does: Changes your current directory to where the application files are located.

```
cd /home/ubuntu/code_artifacts/
```

Verify it worked:

```
ls -lh trading-app-deployment.tar.gz
```

Expected output: You should see something like:

```
-rw-r--r-- 1 ubuntu ubuntu 349K Oct 25 17:45 trading-app-deployment.tar.gz
```

1.3 Copy the Package to Your VPS

What this does: Securely transfers the application files from your computer to the VPS using `scp` (secure copy).

```
scp trading-app-deployment.tar.gz root@67.211.219.94:/root/
```

What happens:
- You might be asked: "Are you sure you want to continue connecting?" → Type `yes` and press Enter
- You'll be prompted for the VPS password → Type it (you won't see characters appear) and press Enter
- You'll see a progress bar as the file transfers

Verify it worked: You should see output like:

```
trading-app-deployment.tar.gz    100%  349KB  1.2MB/s   00:00
```

🔐 STEP 2: Connect to Your VPS

2.1 SSH Into Your VPS

What this does: Logs you into your VPS server so you can run commands on it.

```
ssh root@67.211.219.94
```

What happens:
- Enter your password when prompted
- Your terminal prompt should change to show you're on the VPS (something like `root@vps-name:~#`)

Verify it worked: Run this command:

```
whoami
```

Expected output: `root`

🛠️ STEP 3: Prepare the VPS Environment

3.1 Update System Packages

What this does: Updates the list of available software and installs security updates. This ensures you have the latest versions of everything.

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

This takes: 2-5 minutes

What happens: You'll see lots of text scrolling by as packages are downloaded and installed.

3.2 Install System Dependencies

What this does: Installs tools needed by the application:
- `build-essential` : Compilation tools for building software
- `git` : Version control system
- `curl` & `wget` : Tools for downloading files
- `tesseract-ocr` : Optical character recognition for reading text from images

```
sudo apt-get install -y build-essential git curl wget tesseract-ocr
```

Verify it worked:

```
tesseract --version
```

Expected output: Should show version info like:

```
tesseract 4.1.1
```

3.3 Check if Go is Installed

What this does: Verifies that Go (programming language for the backend) is installed.

```
go version
```

Expected output: Something like:

```
go version go1.21.6 linux/amd64
```

✅ If you see a version number, Go is installed!
❌ If you see "command not found", you need to install Go first.

3.4 Install Node.js and npm

What this does: Installs Node.js (JavaScript runtime) and npm (package manager) needed to run and build the Next.js frontend.

Step 3.4a - Download and run NodeSource setup script:

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
```

What happens: This adds the official Node.js repository to your system.

Step 3.4b - Install Node.js:

```
sudo apt-get install -y nodejs
```

Verify it worked:

```
node --version
npm --version
```

Expected output:

```
v18.20.8
9.8.1
```

(Versions should be 18.x or higher for Node, and 9.x or higher for npm)

📂 STEP 4: Extract and Set Up the Application

4.1 Extract the Deployment Package

What this does: Uncompresses the `.tar.gz` file to extract all application files.

```
cd /root/
tar -xzf trading-app-deployment.tar.gz
```

Verify it worked:

```
ls -la /root/trading-app/
```

Expected output: You should see folders like:

```
backend
frontend
deploy.sh
start.sh
stop.sh
README.md
...
```

4.2 Make Scripts Executable

What this does: Gives execute permission to shell scripts so you can run them.

```
cd /root/trading-app/
chmod +x *.sh
```

Verify it worked:

```
ls -lh *.sh
```

Expected output: All .sh files should start with `-rwxr-xr-x` (the `x` means executable)

⚙️ STEP 5: Configure the Application

5.1 Configure the Backend

What this does: Creates a configuration file that tells the backend how to run.

```
cat > /root/trading-app/backend/.env << 'EOF'
# Database configuration
DB_PATH=./data/trading.db

# Upload directory for files
UPLOAD_DIR=./data/uploads

# Server port (backend will run on this port)
PORT=8080

# OpenAlgo configuration (trading platform)
OPENALGO_URL=https://openalgo.mywire.org
OPENALGO_API_KEY=your_openalgo_api_key_here

# Abacus.AI API key (for AI chat features)
ABACUS_API_KEY=your_abacus_api_key_here

# JWT Secret for authentication (CHANGE THIS!)
JWT_SECRET=change_this_to_a_random_secure_string_123456

# Environment
ENVIRONMENT=production
EOF
```

⚠️ **IMPORTANT**: You need to edit this file and add your real API keys!

Edit the configuration:

```
nano /root/trading-app/backend/.env
```

What to change:
1. Replace `your_openalgo_api_key_here` with your actual OpenAlgo API key

2. Replace `your_abacus_api_key_here` with your actual Abacus.AI API key
3. Replace `change_this_to_a_random_secure_string_123456` with a random secure string

How to save in nano:
- Press `Ctrl + O` (to save)
- Press `Enter` (to confirm)
- Press `Ctrl + X` (to exit)

Verify it worked:

```
cat /root/trading-app/backend/.env
```

Expected output: Should show your configuration with your API keys.

5.2 Configure the Frontend

What this does: Tells the frontend where to find the backend API.

```
cat > /root/trading-app/frontend/.env.local << 'EOF'
# Backend API URL (replace with your VPS IP)
NEXT_PUBLIC_API_URL=http://67.211.219.94:8080

# WebSocket URL for real-time updates
NEXT_PUBLIC_WS_URL=ws://67.211.219.94:8080/ws
EOF
```

Verify it worked:

```
cat /root/trading-app/frontend/.env.local
```

Expected output: Should show the configuration with your VPS IP address.

🔨 STEP 6: Build the Applications

6.1 Build the Go Backend

What this does: Compiles the Go source code into an executable program.

```
cd /root/trading-app/backend/
go mod download
go build -o trading-server ./cmd/main.go
```

This takes: 1-3 minutes (first time may download dependencies)

Verify it worked:

```
ls -lh /root/trading-app/backend/trading-server
```

Expected output: You should see a file around 20-40 MB:

```
-rwxr-xr-x 1 root root 25M Oct 25 18:00 trading-server
```

6.2 Install Frontend Dependencies

What this does: Downloads all the JavaScript packages needed by the Next.js frontend.

```
cd /root/trading-app/frontend/
npm install --legacy-peer-deps
```

This takes: 2-5 minutes

What happens: You'll see lots of packages being downloaded. The `--legacy-peer-deps` flag handles compatibility between packages.

Verify it worked:

```
ls -la /root/trading-app/frontend/node_modules/ | head -20
```

Expected output: Should show many folders (packages).

6.3 Build the Frontend for Production

What this does: Optimizes and prepares the Next.js application for production use.

```
cd /root/trading-app/frontend/
npm run build
```

This takes: 2-5 minutes

What happens: Next.js will analyze your code, optimize it, and create production-ready files.

Verify it worked: You should see output ending with:

```
✓ Compiled successfully
```

And you can check:

```
ls -la /root/trading-app/frontend/.next/
```

Expected output: Should show a `.next` folder with built files.

🚀 STEP 7: Set Up Services to Keep Apps Running

Why we need this: Without this, if you close your SSH session, the applications will stop. We'll use `systemd` to keep them running automatically.

7.1 Create Backend Service

What this does: Creates a service that runs the Go backend and automatically restarts it if it crashes.

```
cat > /etc/systemd/system/trading-backend.service << 'EOF'
[Unit]
Description=Trading App Backend (Go)
After=network.target

[Service]
Type=simple
User=root
WorkingDirectory=/root/trading-app/backend
ExecStart=/root/trading-app/backend/trading-server
Restart=always
RestartSec=10
StandardOutput=append:/var/log/trading-backend.log
StandardError=append:/var/log/trading-backend.log

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

Verify it worked:

```
cat /etc/systemd/system/trading-backend.service
```

Expected output: Should show the service configuration you just created.

7.2 Create Frontend Service

What this does: Creates a service that runs the Next.js frontend.

```
cat > /etc/systemd/system/trading-frontend.service << 'EOF'
[Unit]
Description=Trading App Frontend (Next.js)
After=network.target

[Service]
Type=simple
User=root
WorkingDirectory=/root/trading-app/frontend
ExecStart=/usr/bin/npm start
Restart=always
RestartSec=10
StandardOutput=append:/var/log/trading-frontend.log
StandardError=append:/var/log/trading-frontend.log

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

Verify it worked:

```
cat /etc/systemd/system/trading-frontend.service
```

7.3 Reload systemd and Enable Services

What this does: Tells systemd about the new services and sets them to start automatically on boot.

```
sudo systemctl daemon-reload
sudo systemctl enable trading-backend
sudo systemctl enable trading-frontend
```

Expected output: Should show messages about creating symlinks.

▶️ STEP 8: Start the Applications

8.1 Start the Backend

What this does: Starts the Go backend server.

```
sudo systemctl start trading-backend
```

Wait 3 seconds, then check if it's running:

```
sudo systemctl status trading-backend
```

Expected output: Should show:

```
● trading-backend.service - Trading App Backend (Go)
   Loaded: loaded (/etc/systemd/system/trading-backend.service; enabled)
   Active: active (running) since...
```

✅ **Look for**: `Active: active (running)` in green
❌ **If you see**: `Active: failed` - check logs (see troubleshooting section below)

Press `q` to exit the status view.

8.2 Start the Frontend

What this does: Starts the Next.js frontend server.

```
sudo systemctl start trading-frontend
```

Wait 5 seconds, then check if it's running:

```
sudo systemctl status trading-frontend
```

Expected output: Should show `Active: active (running)`

Press `q` to exit.

🔍 STEP 9: Verify Everything is Working

9.1 Check if Services are Running

```
sudo systemctl status trading-backend --no-pager
sudo systemctl status trading-frontend --no-pager
```

Both should show: `Active: active (running)`

9.2 Check if Ports are Open

What this does: Verifies that the applications are listening on the correct ports.

```
sudo netstat -tulpn | grep -E ':(8080|3000)'
```

Expected output: Should show two lines like:

```
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN      12345/
trading-server
tcp        0      0 0.0.0.0:3000            0.0.0.0:*               LISTEN      12346/
node
```

✅ **Look for**: Port 8080 (backend) and port 3000 (frontend)

If netstat is not installed:

```
sudo apt-get install -y net-tools
```

Then run the check again.

9.3 Test Backend API

What this does: Sends a test request to the backend to see if it responds.

```
curl http://localhost:8080/health
```

Expected output: Should show something like:

```
{"status":"ok"}
```

✅ **If you see JSON response**: Backend is working!
❌ **If you see error**: Check logs (see troubleshooting below)

9.4 Test Frontend

What this does: Checks if the frontend is responding.

```
curl -I http://localhost:3000
```

Expected output: Should start with:

```
HTTP/1.1 200 OK
```

✅ **If you see** `200 OK` : Frontend is working!

9.5 Open in Web Browser

What to do: Open your web browser and go to:

```
http://67.211.219.94:3000
```

Expected result: You should see the trading application login page!

Default login credentials:
- Username: `admin`
- Password: `admin123`

🔥 STEP 10: Configure Firewall (Important for Security!)

What this does: Sets up a firewall to protect your VPS, allowing only necessary ports.

10.1 Install UFW Firewall

```
sudo apt-get install -y ufw
```

10.2 Allow SSH (CRITICAL - Do This First!)

⚠️ **WARNING**: Do NOT skip this step or you'll lock yourself out!

```
sudo ufw allow 22/tcp
```

What this does: Allows SSH connections so you can always access your VPS.

10.3 Allow Application Ports

```
sudo ufw allow 3000/tcp
sudo ufw allow 8080/tcp
```

What this does:
- Port 3000: Allows users to access the frontend website
- Port 8080: Allows the frontend to communicate with the backend API

10.4 Enable the Firewall

```
sudo ufw enable
```

You'll see: "Command may disrupt existing ssh connections. Proceed with operation (y|n)?"

Type: `y` and press Enter

10.5 Check Firewall Status

```
sudo ufw status
```

Expected output:

```
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
3000/tcp                   ALLOW       Anywhere
8080/tcp                   ALLOW       Anywhere
```

📊 Monitoring and Maintenance Commands

View Real-Time Logs

Backend logs:

```
sudo tail -f /var/log/trading-backend.log
```

Frontend logs:

```
sudo tail -f /var/log/trading-frontend.log
```

Press `Ctrl + C` to stop viewing logs.

Restart Services

If you need to restart after making changes:

```
# Restart backend
sudo systemctl restart trading-backend

# Restart frontend
sudo systemctl restart trading-frontend

# Or restart both
sudo systemctl restart trading-backend trading-frontend
```

Stop Services

```
sudo systemctl stop trading-backend
sudo systemctl stop trading-frontend
```

Check System Resources

Memory usage:

```
free -h
```

Disk space:

```
df -h
```

CPU and processes:

```
top
```

(Press `q` to exit)

🆘 Troubleshooting

Problem 1: Backend Won't Start

Check logs:

```
sudo journalctl -u trading-backend -n 50 --no-pager
```

Common issues:
- **"Port already in use"**: Another process is using port 8080
- Fix: `sudo lsof -i :8080` (find the process), then `sudo kill -9 [PID]`
- **"Permission denied"**: Check file permissions
- Fix: `chmod +x /root/trading-app/backend/trading-server`
- **"Database error"**: Database file might be missing
- Fix: `mkdir -p /root/trading-app/backend/data`

Problem 2: Frontend Won't Start

Check logs:

```
sudo journalctl -u trading-frontend -n 50 --no-pager
```

Common issues:
- **"Port 3000 already in use"**: Another process is using port 3000
- Fix: `sudo lsof -i :3000` (find the process), then `sudo kill -9 [PID]`
- **"Module not found"**: Dependencies not installed
- Fix: `cd /root/trading-app/frontend && npm install --legacy-peer-deps`
- **"Build failed"**: Need to rebuild
- Fix: `cd /root/trading-app/frontend && npm run build`

Problem 3: Can't Access from Browser

Check if ports are open:

```
sudo netstat -tulpn | grep -E ':(8080|3000)'
```

Check firewall:

```
sudo ufw status
```

Make sure ports 3000 and 8080 are allowed.

Check if services are running:

```
sudo systemctl status trading-backend trading-frontend
```

Problem 4: Out of Memory

Check memory:

```
free -h
```

Add swap space (virtual memory):

```
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
```

Verify swap is active:

```
free -h
```

You should see swap space listed.

Problem 5: Need to See All Logs

Full backend logs:

```
sudo journalctl -u trading-backend -f
```

Full frontend logs:

```
sudo journalctl -u trading-frontend -f
```

System logs:

```
sudo journalctl -xe
```

🔄 How to Update the Application

If you need to deploy a new version:

1. Stop Services

```
sudo systemctl stop trading-backend trading-frontend
```

2. Backup Current Version

```
cp -r /root/trading-app /root/trading-app.backup-$(date +%Y%m%d)
```

3. Transfer New Files

```
# From your local machine:
scp trading-app-deployment.tar.gz root@67.211.219.94:/root/
```

4. Extract and Build

```
cd /root/
tar -xzf trading-app-deployment.tar.gz
cd /root/trading-app/backend/
go build -o trading-server ./cmd/main.go
cd /root/trading-app/frontend/
npm install --legacy-peer-deps
npm run build
```

5. Start Services

```
sudo systemctl start trading-backend trading-frontend
```

🎉 Success Checklist

Mark each item as you complete it:

- [ ] SSH into VPS successfully
- [ ] Application files transferred and extracted
- [ ] Node.js and npm installed
- [ ] Backend configuration file created with API keys
- [ ] Frontend configuration file created
- [ ] Backend built successfully

- [ ] Frontend dependencies installed
- [ ] Frontend built successfully
- [ ] Backend service created and started
- [ ] Frontend service created and started
- [ ] Both services show “active (running)”
- [ ] Backend API responds to curl test
- [ ] Frontend responds to curl test
- [ ] Can access application in web browser (http://67.211.219.94:3000)
- [ ] Can log in with default credentials
- [ ] Firewall configured and enabled

📞 Need Help?

If you’re stuck:

1. **Check the logs**: Logs usually tell you what went wrong
2. **Verify each step**: Go back through the checklist
3. **Check system resources**: Make sure you have enough memory and disk space
4. **Google the error message**: Often others have solved the same problem
5. **Check the application README**: Located at `/root/trading-app/README.md`

🔐 Security Best Practices (Important!)

After Deployment, Do These ASAP:

1. **Change default login credentials**:
   - Log into the app and change the admin password from `admin123` to something secure

2. **Change JWT secret**:
   - Edit `/root/trading-app/backend/.env`
   - Change `JWT_SECRET` to a long random string
   - Restart backend: `sudo systemctl restart trading-backend`

3. **Keep your API keys secret**:
   - Never share your `.env` files
   - Never commit them to public repositories

4. **Regular updates**:
   ```
   bash
     sudo apt-get update && sudo apt-get upgrade -y
   ```

5. **Backup your database regularly**:
   ```
   bash
     mkdir -p /root/backups
     cp /root/trading-app/backend/data/trading.db /root/backups/trading-$(date +%Y%m%d).db
   ```

🎯 Quick Command Reference

Most Common Commands

Task	Command
Check service status	`sudo systemctl status trading-backend trading-frontend`
Restart services	`sudo systemctl restart trading-backend trading-frontend`
View backend logs	`sudo tail -f /var/log/trading-backend.log`
View frontend logs	`sudo tail -f /var/log/trading-frontend.log`
Check what's using ports	`sudo netstat -tulpn | grep -E ':(8080|3000)'`
Check memory	`free -h`
Check disk space	`df -h`
View firewall rules	`sudo ufw status`

✅ You're Done!

If you've completed all steps and everything is working:

🎊 **Congratulations!** Your trading application is now live at:

🌐 http://67.211.219.94:3000

Login with:
- Username: `admin`
- Password: `admin123`

Don't forget to:
- Change the default password
- Add your real API keys
- Set up regular backups

Happy Trading! 📈💰