🚀 Allice Platform - Complete Deployment Guide

Table of Contents

Overview

Allice Platform is a fully autonomous AI agent system with advanced capabilities including:

- 🤖 Multiple specialized AI agents
- 🔍 Web scraping and automation
- 📱 Social media integration
- ☁️ Cloud infrastructure management
- 🔐 Secure API with authentication
- 📊 Admin dashboard
- 🐘 PostgreSQL database

Quick Start

For the Impatient

Unix/Linux/macOS:

```
chmod +x quick-install.sh
./quick-install.sh
```

Windows:

```
Right-click quick-install.bat → Run as Administrator
```

That’s it! The quick installer will guide you through the process.

Deployment Options

1. Docker Compose (Recommended)

Best for: Quick setup, development, and production deployments

Prerequisites

- Docker Desktop or Docker Engine
- docker-compose

Steps

1. **Ensure you have the deployment package**
   ```
   bash
   cd ailice-deployment-package
   ```
2. **Create environment configuration**
   ```
   bash
   cp .env.example .env
   # Edit .env with your settings
   ```
3. **Start services**
   ```
   bash
   chmod +x docker-compose-start.sh
   ./docker-compose-start.sh
   ```

Or manually:

```
bash
docker-compose up -d
```

1. **Access the platform**
   - Home: http://localhost:8080
   - Admin Dashboard: http://localhost:8080/admin/dashboard
   - API Docs: http://localhost:8080/docs

Management Commands

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

# Stop services
docker-compose down

# Restart services
docker-compose restart

# View service status
docker-compose ps

# Access database
docker-compose exec postgres psql -U ailice -d ailice_db
```

Including PgAdmin

To start with database management UI:

```
docker-compose --profile admin up -d
```

Access PgAdmin at: http://localhost:5050

---

2. Google Cloud Run

Best for: Production deployments, scalability, managed infrastructure

Prerequisites

- Google Cloud account
- gcloud CLI installed
- Project ID: `eighth-beacon-479707-c3`

Automated Deployment

```
chmod +x deploy-to-gcloud.sh
./deploy-to-gcloud.sh
```

The script will:
1. ✅ Check gcloud CLI installation
2. ✅ Authenticate with Google Cloud
3. ✅ Enable required APIs
4. ✅ Build Docker image
5. ✅ Create Cloud SQL PostgreSQL instance (optional)
6. ✅ Deploy to Cloud Run
7. ✅ Configure environment variables
8. ✅ Provide deployment URL

What Gets Created

- **Cloud Run Service**: `viralspark-ailice`
- **Container Registry**: Image stored in GCR
- **Cloud SQL Instance**: `ailice-postgres` (if selected)
- **Secret Manager**: Database credentials

Cost Optimization

- Cloud Run: Pay-per-use (scales to zero)
- Cloud SQL: `db-f1-micro` tier (upgradable)
- Estimated cost: $5-20/month for light usage

Manual Steps

If you prefer manual deployment:

```
# Authenticate
gcloud auth login
gcloud config set project eighth-beacon-479707-c3

# Build and push image
gcloud builds submit --tag gcr.io/eighth-beacon-479707-c3/viralspark-ailice

# Deploy to Cloud Run
gcloud run deploy viralspark-ailice \
  --image gcr.io/eighth-beacon-479707-c3/viralspark-ailice \
  --region us-central1 \
  --platform managed \
  --allow-unauthenticated \
  --memory 2Gi
```

---

3. Hostinger VPS

Best for: Budget-friendly hosting, full control

Prerequisites

- Hostinger VPS or Cloud hosting plan
- Ubuntu 20.04+ recommended
- SSH access

Automated Deployment

1. **Upload deployment package to your VPS**
   `bash`
   `scp -r ailice-deployment-package root@your-vps-ip:/root/`

2. **SSH into your VPS**
   `bash`
   `ssh root@your-vps-ip`

3. **Run deployment script**
   `bash`
   `cd /root/ailice-deployment-package`
   `chmod +x deploy-to-hostinger.sh`
   `./deploy-to-hostinger.sh`

The script will:
1. ✅ Update system packages
2. ✅ Install dependencies (Python, PostgreSQL, Nginx)
3. ✅ Configure PostgreSQL database
4. ✅ Choose deployment method (Docker or Native)
5. ✅ Set up Nginx reverse proxy
6. ✅ Configure firewall
7. ✅ Install SSL certificate (Let's Encrypt)

Deployment Methods

Option 1: Docker (Recommended)
- Isolated environment

- Easy updates
- Better resource management

Option 2: Native Python
- Direct control
- Systemd service
- Slightly faster

Domain Setup

Point your domain to your VPS IP:

```
A Record: @ → your-vps-ip
A Record: www → your-vps-ip
```

---

4. macOS Local Installation

Best for: Development, local testing on Mac

Prerequisites

- macOS 10.15 or later
- Admin access

Automated Installation

```
chmod +x install-mac.sh
./install-mac.sh
```

The script will:
1. ✅ Install Homebrew (if needed)
2. ✅ Install Python 3.11
3. ✅ Install PostgreSQL 15
4. ✅ Create virtual environment
5. ✅ Install dependencies
6. ✅ Set up database
7. ✅ Create launch scripts
8. ✅ Configure auto-start (optional)
9. ✅ Start the service

Manual Start/Stop

```
# Start AIlice
cd ~/AIlice
./start_ailice.sh

# Stop AIlice
cd ~/AIlice
./stop_ailice.sh

# View logs
tail -f ~/AIlice/logs/ailice.log
```

Auto-start on Login

The installer will ask if you want auto-start. If you skipped it:

```
cp ~/AIlice/com.ailice.platform.plist ~/Library/LaunchAgents/
launchctl load ~/Library/LaunchAgents/com.ailice.platform.plist
```

---

5. Windows Local Installation

Best for: Development, local testing on Windows

Prerequisites

- Windows 10/11
- Administrator access

Automated Installation

1. **Right-click** `install-windows.ps1`
2. **Select "Run with PowerShell"**
3. **Approve Administrator prompt**

The script will:
1. ✅ Install Chocolatey package manager
2. ✅ Install Python 3.11
3. ✅ Install PostgreSQL 15
4. ✅ Install Git
5. ✅ Create virtual environment
6. ✅ Install dependencies
7. ✅ Set up database
8. ✅ Create launch scripts
9. ✅ Create desktop shortcut
10. ✅ Configure Windows service (optional)

Manual Start/Stop

Using Desktop Shortcut:
- Double-click "Allice Platform" on desktop

Using Command Line:

```
cd %USERPROFILE%\AIlice
start_ailice.bat
```

Stop:

```
cd %USERPROFILE%\AIlice
stop_ailice.bat
```

Windows Service

If you created a Windows service:

```
# Start service
net start AIlicePlatform

# Stop service
net stop AIlicePlatform

# Check status
sc query AIlicePlatform
```

Configuration

Environment Variables

Create a `.env` file or set environment variables:

```
# Database
DATABASE_URL=postgresql://ailice:password@localhost:5432/ailice_db

# Application
ENVIRONMENT=production
PORT=8080
HOST=0.0.0.0

# CORS (comma-separated)
CORS_ORIGINS=*

# API Keys (optional)
OPENAI_API_KEY=sk-...
ANTHROPIC_API_KEY=sk-ant-...
GOOGLE_API_KEY=AIza...

# Social Media (optional)
TWITTER_API_KEY=...
TWITTER_API_SECRET=...

# Cloud Providers (optional)
AWS_ACCESS_KEY_ID=...
AWS_SECRET_ACCESS_KEY=...
GCP_PROJECT_ID=...
```

Model Configuration

Edit `config.json` to configure AI models:

```
{
  "modelID": "",
  "agentModelConfig": {
    "default": "ollama:gemma3:27b",
    "researcher": "cloud:deepseek-v3.1:671b-cloud",
    "coder": "ollama:qwen3-coder:30b"
  },
  "models": {
    "ollama": {
      "baseURL": "http://localhost:11434/v1"
    }
  }
}
```

Capabilities Configuration

Edit `capabilities_config.json` to enable/disable features:

```
{
  "realWorldCapabilities": {
    "web_automation": {
      "enabled": true
    },
    "social_media": {
      "enabled": true,
      "platforms": ["twitter", "linkedin"]
    }
  }
}
```

Troubleshooting

Common Issues

Port Already in Use

```
# Find process using port 8080
lsof -i :8080          # macOS/Linux
netstat -ano | findstr :8080  # Windows

# Kill the process or change PORT in .env
```

Database Connection Failed

```
# Check if PostgreSQL is running
# macOS/Linux
ps aux | grep postgres

# Windows
sc query postgresql-x64-15

# Test connection
psql -U ailice -d ailice_db -h localhost
```

Docker Issues

```
# Rebuild containers
docker-compose down -v
docker-compose build --no-cache
docker-compose up -d

# Check logs
docker-compose logs -f ailice
```

Python Dependency Conflicts

```
# Recreate virtual environment
rm -rf venv
python3 -m venv venv
source venv/bin/activate
pip install --upgrade pip
pip install -r requirements.txt
```

Getting Help

1. **Check logs:**
   - Docker: `docker-compose logs -f`
   - Native: `tail -f logs/ailice.log`
   - Systemd: `journalctl -u ailice -f`

2. **Verify environment:**
   ```bash
   # Check Python version
   python –version # Should be 3.10+

# Check PostgreSQL
psql –version # Should be 15+

# Check dependencies
pip list
```

1. **Database diagnostics:**
   ```bash
   # Check database exists
   psql -U postgres -l | grep ailice_db

# Check user permissions
psql -U postgres -c “\du ailice”
```

Performance Optimization

Production Settings

1. **Use production WSGI server:**
   - Already configured with Uvicorn
   - Increase workers: `uvicorn.run(workers=4)`

2. **Database optimization:**
   ```sql
   – Create indexes
   CREATE INDEX idx_created_at ON your_table(created_at);

– Vacuum regularly
VACUUM ANALYZE;
```

1. **Caching:**
   - Add Redis for caching
   - Configure in `docker-compose.yml`

2. **Resource limits:**
   ```yaml
   # docker-compose.yml
   services:
     ailice:
       deploy:
         resources:
           limits:
             cpus: '2'
             memory: 4G
   ```

Monitoring

1. **Health checks:**
   - Endpoint: `http://localhost:8080/health`
   - Set up monitoring alerts

2. **Logs:**
   - Centralize logs with ELK stack
   - Or use Cloud Logging (GCP)

3. **Metrics:**
   - Prometheus endpoint at `/metrics`
   - Grafana dashboards available

Security Best Practices

1. **Use strong passwords:**
   ```bash
   # Generate secure password
   openssl rand -base64 32
   ```

2. **Enable HTTPS:**
   - Use Let's Encrypt (free)
   - Hostinger script includes SSL setup
   - Cloud Run provides HTTPS by default

3. **Firewall configuration:**
   ```bash
   # Allow only necessary ports
   ufw allow 80/tcp
   ufw allow 443/tcp
   ufw allow 22/tcp
   ufw enable
   ```

4. **Environment variables:**
   - Never commit `.env` to git
   - Use secret management in production

5. **Regular updates:**
   ```bash
   # Update system packages
   apt update && apt upgrade -y

# Update Python packages
pip install –upgrade -r requirements.txt
```

Backup and Recovery

Database Backup

```bash
# Backup
pg_dump -U ailice ailice_db > backup_$(date +%Y%m%d).sql

# Restore
psql -U ailice ailice_db < backup_20240101.sql

# Docker backup
docker-compose exec postgres pg_dump -U ailice ailice_db > backup.sql
```

Automated Backups

```bash
# Add to crontab
0 2 * * * /path/to/backup_script.sh
```

Full Backup

```
# Backup everything
tar -czf ailice_backup_$(date +%Y%m%d).tar.gz \
  ~/AIlice/ \
  --exclude='venv' \
  --exclude='__pycache__'
```

Scaling

Horizontal Scaling

1. **Load balancer:**
   - Nginx, HAProxy, or cloud load balancer
   - Multiple AIlice instances

2. **Database replication:**
   - Primary-replica setup
   - Read replicas for scaling reads

3. **Caching layer:**
   - Redis for session management
   - CDN for static assets

Vertical Scaling

1. **Increase resources:**
   - More CPU/RAM
   - Faster storage (SSD)

2. **Optimize code:**
   - Async operations
   - Connection pooling
   - Query optimization

Support

Documentation

- Main README: `README.md`
- API Documentation: http://localhost:8080/docs
- Configuration guide: `config.json`

Community

- GitHub Issues: Report bugs and feature requests
- Discussions: Ask questions and share ideas

Commercial Support

For enterprise support, custom development, or consulting:
- Email: support@ailice.platform
- Website: https://ailice.platform

License

Please refer to the LICENSE file in the repository.

Changelog

Version 1.0.0

- Initial release
- Multiple deployment options
- Docker support
- Cloud deployment ready
- Admin dashboard
- API documentation

Thank you for using AIlice Platform! 🚀