Go-Ichiran API Server - Complete Deployment Guide

This guide provides step-by-step instructions for deploying the Go-Ichiran API server to an Ubuntu 22.04 LTS VPS.

📁 Project Structure

Your deployment package includes the following files:

```
go-ichiran-api/
├── README.md                    # Main documentation
├── DEPLOYMENT_GUIDE.md          # This deployment guide
├── docker-compose.yaml          # Multi-service orchestration
├── Dockerfile                   # Go API server container
├── Dockerfile.ichiran           # Ichiran Common Lisp container
├── nginx.conf                   # Reverse proxy configuration
├── deploy-to-vps.sh            # Automated deployment script
├── test-api.sh                 # API testing script
├── code/                       # Go API server source code
│   ├── main.go                 # Main API server application
│   └── go.mod                  # Go module dependencies
├── scripts/                    # Setup and configuration scripts
│   ├── setup-ichiran.lisp      # Ichiran initialization script
│   ├── settings.lisp           # Ichiran configuration
│   ├── ichiran-cli.sh          # CLI wrapper for ichiran
│   └── wait-for-db.sh          # Database readiness script
└── examples/                   # Usage examples and documentation
    └── api-examples.md         # Comprehensive API usage examples
```

🚀 Quick Deployment (Recommended)

Option 1: Automated Deployment

1. **Upload files to your VPS:**
   `bash # From your local machine tar -czf go-ichiran-api.tar.gz go-ichiran-api/ scp go-ichiran-api.tar.gz user@your-vps:/home/user/`
2. **Connect to VPS and extract:**
   `bash ssh user@your-vps tar -xzf go-ichiran-api.tar.gz cd go-ichiran-api`
3. **Run automated deployment:**
   `bash chmod +x deploy-to-vps.sh ./deploy-to-vps.sh`
4. **Start the application:**
   `bash cd /opt/go-ichiran-api chmod +x deploy.sh ./deploy.sh start`
5. **Test the deployment:**
   `bash chmod +x test-api.sh ./test-api.sh`

Option 2: Manual Deployment

If you prefer manual control over the installation process:

🔧 Manual Installation Steps

Step 1: System Preparation

```
# Update system
sudo apt-get update && sudo apt-get upgrade -y

# Install essential packages
sudo apt-get install -y curl wget git unzip software-properties-common
```

Step 2: Install Docker

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

# Install Docker Compose
sudo curl -L "https://github.com/docker/compose/releases/latest/
download/docker-compose-<span class="math-inline" style="display:
inline;"><math xmlns="http://www.w3.org/1998/Math/MathML"
display="inline"><mrow><mo stretchy="false">&#x00028;</mo><mi>u</
mi><mi>n</mi><mi>a</mi><mi>m</mi><mi>e</mi><mo>&#x02212;</
mo><mi>s</mi><mo stretchy="false">&#x00029;</mo><mo>&#x02212;</
mo></mrow></math></span>(uname -m)" -o /usr/local/bin/docker-
compose
sudo chmod +x /usr/local/bin/docker-compose

# Log out and back in for group changes to take effect
```

Step 3: Install Go (Optional, for development)

```
GO_VERSION="1.21.5"
cd /tmp
wget https://go.dev/dl/go${GO_VERSION}.linux-amd64.tar.gz
sudo rm -rf /usr/local/go
sudo tar -C /usr/local -xzf go${GO_VERSION}.linux-amd64.tar.gz
echo 'export PATH=$PATH:/usr/local/go/bin' >> ~/.bashrc
source ~/.bashrc
```

Step 4: Deploy Application

```
# Create application directory
sudo mkdir -p /opt/go-ichiran-api
sudo chown <span class="math-inline" style="display:
inline;"><math xmlns="http://www.w3.org/1998/Math/MathML"
display="inline"><mrow><mi>U</mi><mi>S</mi><mi>E</mi><mi>R</
mi><mi>:</mi></mrow></math></span>USER /opt/go-ichiran-api

# Copy application files
cp -r go-ichiran-api/* /opt/go-ichiran-api/
cd /opt/go-ichiran-api

# Make scripts executable
chmod +x scripts/*.sh deploy-to-vps.sh test-api.sh

# Create environment file
cat > .env << EOF
COMPOSE_PROJECT_NAME=go-ichiran-api
POSTGRES_DB=ichiran
POSTGRES_USER=ichiran
POSTGRES_PASSWORD=$(openssl rand -base64 32)
PORT=8080
GIN_MODE=release
EOF
```

Step 5: Start Services

```
# Start all services
docker-compose up -d

# Check service status
docker-compose ps

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

🧪 Testing Your Deployment

Basic Health Check

```
# Wait for services to start (may take 2-3 minutes)
sleep 120

# Test health endpoint
curl http://localhost:8080/health
```

Comprehensive Test Suite

```
# Run complete test suite
./test-api.sh
```

Manual API Testing

```
# Test text analysis
curl -X POST http://localhost:8080/api/v1/analyze \
  -H "Content-Type: application/json" \
  -d '{"text": "こんにちは"}'

# Test romanization
curl -X POST http://localhost:8080/api/v1/romanize \
  -H "Content-Type: application/json" \
  -d '{"text": "一覧は最高だぞ"}'

# Test kanji analysis
curl -X POST http://localhost:8080/api/v1/kanji \
  -H "Content-Type: application/json" \
  -d '{"kanji": "漢"}'
```

🌐 Production Configuration

Configure Domain and SSL

1. **Point your domain to the VPS IP address**
2. **Install SSL certificate (Let's Encrypt):**
   `bash sudo apt install certbot sudo certbot certonly --standalone -d yourdomain.com`
3. **Update nginx.conf:**
   `bash # Edit nginx.conf to enable HTTPS # Uncomment SSL server block # Update server_name with your domain`
4. **Restart services:**
   `bash docker-compose restart nginx`

Security Configuration

1. **Configure firewall:**
   `bash sudo ufw allow 22,80,443/tcp sudo ufw enable`
2. **Update default passwords:**
   `bash # Edit .env file and change database password nano .env`
   `docker-compose down docker-compose up -d`

Performance Optimization

1. **Scale API servers:**
   `bash docker-compose up -d --scale api=3`
2. **Monitor resource usage:**
   `bash ./monitor.sh`

📊 Management Commands

The deployment includes convenient management scripts:

```
# Start services
./deploy.sh start

# Stop services
./deploy.sh stop

# Restart services
./deploy.sh restart

# View logs
./deploy.sh logs

# Check status
./deploy.sh status

# Update and rebuild
./deploy.sh update

# Monitor system
./monitor.sh
```

🔍 Troubleshooting

Common Issues and Solutions

1. Services fail to start:

```
# Check Docker status
sudo systemctl status docker

# Check container logs
docker-compose logs ichiran
docker-compose logs api
```

2. Database connection issues:

```
# Check PostgreSQL health
docker-compose exec postgres pg_isready -U ichiran

# Reset database
docker-compose down -v
docker-compose up -d
```

3. Ichiran initialization problems:

```
# Check ichiran container
docker exec -it ichiran-container bash
ichiran-cli --help

# Restart ichiran service
docker-compose restart ichiran
```

4. API returns errors:

```
# Test ichiran directly
docker exec ichiran-container ichiran-cli -i "テスト"

# Check API logs
docker-compose logs api
```

Performance Issues

High memory usage:

```
# Monitor resource usage
docker stats

# Adjust container memory limits in docker-compose.yaml
```

Slow response times:

```
# Check if ichiran needs warmup time
# The first few requests may be slower

# Monitor response times
time curl -X POST http://localhost:8080/api/v1/analyze \
  -H "Content-Type: application/json" \
  -d '{"text": "テスト"}'
```

📚 Additional Resources

- **Main Documentation**: README.md
- **API Examples**: examples/api-examples.md
- **Ichiran Library**: GitHub Repository
- **Docker Documentation**: docs.docker.com

🆘 Support

If you encounter issues:

1. **Check logs first**: `docker-compose logs`
2. **Run test suite**: `./test-api.sh`

3. **Check service health**: `docker-compose ps`
4. **Monitor resources**: `./monitor.sh`
5. **Review troubleshooting section** in this guide

📝 Final Checklist

Before considering your deployment complete:

- [ ] All containers are running (`docker-compose ps`)
- [ ] Health check passes (`curl http://localhost:8080/health`)
- [ ] Test suite passes (`./test-api.sh`)
- [ ] API endpoints respond correctly
- [ ] SSL certificate configured (for production)
- [ ] Firewall configured
- [ ] Monitoring set up
- [ ] Backup strategy implemented

Deployment completed successfully! 🎉

Your Go-Ichiran API server is now ready to serve Japanese text analysis requests.

Quick Test:

```
curl -X POST http://your-server/api/v1/analyze \
  -H "Content-Type: application/json" \
  -d '{"text": "ありがとうございます"}'
```

Author: MiniMax Agent
Version: 1.0.0
Last Updated: 2025-06-22