Adityzer Setup Guide

Quick Start

Prerequisites

- Node.js >= 16.0.0
- npm or yarn
- Git
- Docker (optional)

Installation

```
# Clone the repository
git clone https://github.com/Adityzer/adityzer.git
cd adityzer

# Install dependencies
npm install

# Run setup wizard
npm run setup

# Start the application
npm start
```

Detailed Setup

1. Environment Configuration

Create a `.env` file in the project root:

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

# Edit configuration
nano .env
```

Required environment variables:

```
# Core Configuration
NODE_ENV=development
PORT=3000
API_KEY=your-api-key-here

# Blockchain Networks
ETHEREUM_RPC_URL=https://mainnet.infura.io/v3/your-infura-key
POLYGON_RPC_URL=https://polygon-mainnet.infura.io/v3/your-infura-key
ARBITRUM_RPC_URL=https://arb1.arbitrum.io/rpc
OPTIMISM_RPC_URL=https://mainnet.optimism.io

# AI Configuration
OPENAI_API_KEY=your-openai-api-key
AI_MODEL=gpt-4

# Database
DATABASE_URL=postgresql://user:password@localhost:5432/audityzer

# Monitoring
PROMETHEUS_ENDPOINT=http://localhost:9090
GRAFANA_ENDPOINT=http://localhost:3001
```

2. Database Setup

PostgreSQL (Recommended)

```
# Install PostgreSQL
sudo apt-get install postgresql postgresql-contrib

# Create database
sudo -u postgres createdb audityzer

# Create user
sudo -u postgres createuser --interactive audityzer

# Set password
sudo -u postgres psql -c "ALTER USER audityzer PASSWORD 'your-password';"

# Grant privileges
sudo -u postgres psql -c "GRANT ALL PRIVILEGES ON DATABASE audityzer TO audityzer;"
```

SQLite (Development)

```
# SQLite is included by default for development
# No additional setup required
```

3. Blockchain Network Configuration

Infura Setup

1. Create account at Infura (https://infura.io)
2. Create new project
3. Copy project ID
4. Add to `.env` file

Alchemy Setup (Alternative)

1. Create account at Alchemy (https://alchemy.com)
2. Create new app
3. Copy API key
4. Update RPC URLs in `.env`

Local Node Setup

```
# Ethereum (Hardhat)
npx hardhat node

# Or Ganache
npm install -g ganache-cli
ganache-cli
```

4. AI Configuration

OpenAI Setup

1. Create account at OpenAI (https://openai.com)
2. Generate API key
3. Add to `.env` file

Local AI Models (Optional)

```
# Install Ollama
curl -fsSL https://ollama.ai/install.sh | sh

# Download models
ollama pull llama2
ollama pull codellama

# Update configuration
AI_PROVIDER=ollama
AI_MODEL=llama2
OLLAMA_ENDPOINT=http://localhost:11434
```

5. Docker Setup

Using Docker Compose

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

# View logs
docker-compose logs -f

# Stop services
docker-compose down
```

Custom Docker Configuration

```yaml
# docker-compose.override.yml
version: '3.8'
services:
  audityzer:
    environment:
      - NODE_ENV=development
      - DEBUG=audityzer:*
    volumes:
      - ./custom-config:/app/config
    ports:
      - "3000:3000"
      - "9229:9229" # Debug port
```

6. Monitoring Setup

Prometheus

```
# Install Prometheus
wget https://github.com/prometheus/prometheus/releases/download/v2.40.0/
prometheus-2.40.0.linux-amd64.tar.gz
tar xvfz prometheus-*.tar.gz
cd prometheus-*

# Configure Prometheus
cp prometheus.yml prometheus.yml.backup
cat > prometheus.yml << EOF
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'audityzer'
    static_configs:
      - targets: ['localhost:3000']
EOF

# Start Prometheus
./prometheus --config.file=prometheus.yml
```

Grafana

```
# Install Grafana
sudo apt-get install -y software-properties-common
sudo add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
sudo apt-get update
sudo apt-get install grafana

# Start Grafana
sudo systemctl start grafana-server
sudo systemctl enable grafana-server

# Access Grafana at http://localhost:3000
# Default credentials: admin/admin
```

Configuration Options

Core Configuration

```
// audityzer.config.js
module.exports = {
  // Network configuration
  networks: {
    ethereum: {
      rpc: process.env.ETHEREUM_RPC_URL,
      chainId: 1,
      gasPrice: 'auto'
    },
    polygon: {
      rpc: process.env.POLYGON_RPC_URL,
      chainId: 137,
      gasPrice: 'auto'
    }
  },

  // AI configuration
  ai: {
    provider: 'openai',
    model: 'gpt-4',
    temperature: 0.1,
    maxTokens: 2048
  },

  // Security configuration
  security: {
    enableVulnerabilityScanning: true,
    enableThreatDetection: true,
    scanInterval: 300000, // 5 minutes
    alertThreshold: 'medium'
  },

  // Bridge configuration
  bridges: {
    layerzero: {
      enabled: true,
      networks: ['ethereum', 'polygon', 'arbitrum']
    },
    stargate: {
      enabled: true,
      networks: ['ethereum', 'polygon', 'arbitrum']
    }
  },

  // Dashboard configuration
  dashboard: {
    enabled: true,
    port: 3001,
    realTimeUpdates: true,
    refreshInterval: 5000
  }
};
```

Advanced Configuration

```
// config/advanced.js
module.exports = {
  // Performance tuning
  performance: {
    maxConcurrentScans: 10,
    requestTimeout: 30000,
    retryAttempts: 3,
    cacheEnabled: true,
    cacheTTL: 300000
  },

  // Logging configuration
  logging: {
    level: 'info',
    format: 'json',
    outputs: ['console', 'file'],
    file: {
      path: './logs/audityzer.log',
      maxSize: '10m',
      maxFiles: 5
    }
  },

  // Rate limiting
  rateLimit: {
    enabled: true,
    windowMs: 60000, // 1 minute
    maxRequests: 100,
    skipSuccessfulRequests: false
  },

  // Authentication
  auth: {
    enabled: false,
    provider: 'oauth2',
    config: {
      clientId: process.env.AUTH_CLIENT_ID,
      clientSecret: process.env.AUTH_CLIENT_SECRET
    }
  }
};
```

Testing Setup

Unit Tests

```
# Install test dependencies
npm install --save-dev jest supertest

# Run tests
npm test

# Run with coverage
npm run test:coverage

# Watch mode
npm run test:watch
```

Integration Tests

```
# Setup test database
createdb audityzer_test

# Run integration tests
npm run test:integration

# Run specific test suite
npm run test:integration -- --grep "bridge"
```

E2E Tests

```
# Install Playwright
npx playwright install

# Run E2E tests
npm run test:e2e

# Run with UI
npm run test:e2e -- --ui

# Run specific browser
npm run test:e2e -- --project=chromium
```

Development Setup

IDE Configuration

VS Code

```
// .vscode/settings.json
{
  "editor.formatOnSave": true,
  "editor.codeActionsOnSave": {
    "source.fixAll.eslint": true
  },
  "eslint.workingDirectories": ["./"],
  "typescript.preferences.importModuleSpecifier": "relative"
}
```

Extensions

- ESLint
- Prettier
- TypeScript and JavaScript Language Features
- GitLens
- Docker

Git Hooks

```
# Install husky
npm install --save-dev husky

# Setup pre-commit hooks
npx husky install
npx husky add .husky/pre-commit "npm run lint && npm test"
npx husky add .husky/commit-msg "npx commitlint --edit $1"
```

Debugging

```
# Debug mode
npm run dev:debug

# VS Code debug configuration
# .vscode/launch.json
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Debug Adudityzer",
      "type": "node",
      "request": "launch",
      "program": "${workspaceFolder}/src/index.js",
      "env": {
        "NODE_ENV": "development",
        "DEBUG": "audityzer:*"
      }
    }
  ]
}
```

Production Deployment

Server Requirements

- CPU: 4+ cores
- RAM: 8GB+ recommended
- Storage: 100GB+ SSD
- Network: Stable internet connection

Deployment Steps

```
# 1. Clone repository
git clone https://github.com/Audityzer/audityzer.git
cd audityzer

# 2. Install dependencies
npm ci --only=production

# 3. Build application
npm run build

# 4. Setup environment
cp .env.example .env
# Edit .env with production values

# 5. Setup database
npm run db:migrate

# 6. Start application
npm run start:prod

# Or with PM2
npm install -g pm2
pm2 start ecosystem.config.js --env production
```

Nginx Configuration

```
# /etc/nginx/sites-available/audityzer
server {
    listen 80;
    server_name your-domain.com;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        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;
        proxy_cache_bypass $http_upgrade;
    }
}
```

SSL Setup

```
# Install Certbot
sudo apt-get install certbot python3-certbot-nginx

# Get SSL certificate
sudo certbot --nginx -d your-domain.com

# Auto-renewal
sudo crontab -e
# Add: 0 12 * * * /usr/bin/certbot renew --quiet
```

Troubleshooting

Common Issues

Port Already in Use

```
# Find process using port
lsof -i :3000

# Kill process
kill -9 <PID>

# Or use different port
PORT=3001 npm start
```

Database Connection Issues

```
# Check PostgreSQL status
sudo systemctl status postgresql

# Check connection
psql -h localhost -U audityzer -d audityzer

# Reset password
sudo -u postgres psql -c "ALTER USER audityzer PASSWORD 'newpassword';"
```

RPC Connection Issues

```
# Test RPC connection
curl -X POST \
  -H "Content-Type: application/json" \
  -d '{"jsonrpc":"2.0","method":"eth_blockNumber","params":[],"id":1}' \
  $ETHEREUM_RPC_URL
```

Memory Issues

```
# Increase Node.js memory limit
node --max-old-space-size=4096 src/index.js

# Or set environment variable
export NODE_OPTIONS="--max-old-space-size=4096"
```

Performance Optimization

Database Optimization

```
-- Create indexes
CREATE INDEX idx_vulnerabilities_created_at ON vulnerabilities(created_at);
CREATE INDEX idx_transactions_hash ON transactions(hash);

-- Analyze tables
ANALYZE vulnerabilities;
ANALYZE transactions;
```

Caching

```
// Enable Redis caching
const redis = require('redis');
const client = redis.createClient({
  host: 'localhost',
  port: 6379
});

// Cache configuration
const cacheConfig = {
  ttl: 300, // 5 minutes
  checkperiod: 600 // 10 minutes
};
```

Monitoring

Health Checks

```
# Application health
curl http://localhost:3000/health

# Database health
curl http://localhost:3000/health/db

# External services health
curl http://localhost:3000/health/external
```

Log Analysis

```
# View application logs
tail -f logs/audityzer.log

# Search for errors
grep -i error logs/audityzer.log

# Monitor in real-time
tail -f logs/audityzer.log | grep -i "vulnerability\|threat"
```

Support

Getting Help

- **Documentation**: https://docs.audityzer.com (https://docs.audityzer.com)
- **Discord**: Join our Discord (https://discord.gg/audityzer)
- **GitHub Issues**: Report bugs (https://github.com/Audityzer/audityzer/issues)
- **Email**: support@audityzer.com

Contributing

See our Contributing Guide (CONTRIBUTING.md) for information on how to contribute to the project.

Need help? Join our Discord community (https://discord.gg/audityzer) for real-time support!