

CasperVPN Local Development Guide

Table of Contents

- [Quick Start](#)
- [Prerequisites](#)
- [Initial Setup](#)
- [Running the Application](#)
- [Development Workflow](#)
- [Database Management](#)
- [Testing](#)
- [Debugging](#)
- [Common Tasks](#)
- [IDE Configuration](#)

Quick Start

Get CasperVPN running locally in 5 minutes:

```
# 1. Clone repository
git clone https://github.com/yourusername/caspervpn.git
cd caspervpn

# 2. Initial setup
./scripts/setup.sh

# 3. Start development environment
./scripts/deploy.sh dev

# 4. Verify everything is running
./scripts/health-check.sh
```

✅ Done! Access services at:

- **React Admin:** <http://localhost:3000>
- **API:** <http://localhost:8080>
- **PHP Admin:** <http://localhost:9000>
- **Server Agent:** <http://localhost:8081>

Prerequisites

Required Software

1. Docker Desktop

- [Download for Mac](https://www.docker.com/products/docker-desktop/) (<https://www.docker.com/products/docker-desktop/>)
- [Download for Windows](https://www.docker.com/products/docker-desktop/) (<https://www.docker.com/products/docker-desktop/>)
- [Download for Linux](https://docs.docker.com/desktop/install/linux-install/) (<https://docs.docker.com/desktop/install/linux-install/>)

2. Git

```
```bash
```

```
Mac (with Homebrew)
brew install git

Ubuntu/Debian
sudo apt-get install git

Windows
Download from https://git-scm.com/download/win
...
```

#### 1. **Code Editor** (Choose one)

- [Visual Studio Code](https://code.visualstudio.com/) (https://code.visualstudio.com/) (Recommended)
- [JetBrains Rider](https://www.jetbrains.com/rider/) (https://www.jetbrains.com/rider/) (for .NET)
- [PHPStorm](https://www.jetbrains.com/phpstorm/) (https://www.jetbrains.com/phpstorm/) (for PHP)

## Optional Tools

```
Postman (API testing)
Download from https://www.postman.com/downloads/

TablePlus (Database GUI)
Download from https://tableplus.com/

Docker Extension for VS Code
code --install-extension ms-azuretools.vscode-docker
```

## Initial Setup

### 1. Clone Repository

```
git clone https://github.com/yourusername/caspervpn.git
cd caspervpn
```

### 2. Run Setup Script

```
chmod +x scripts/*.sh
./scripts/setup.sh
```

This will:

- ✓ Check Docker installation
- ✓ Create `.env` file
- ✓ Generate SSL certificates
- ✓ Build Docker images
- ✓ Initialize database
- ✓ Add local hosts

### 3. Configure Environment

The setup script creates `.env` from `.env.example`. For development, defaults are fine.

**Optional:** Customize `.env` if needed:

```
nano .env
```

## 4. Start Development Environment

```
./scripts/deploy.sh dev
```

Wait for services to start (about 30-60 seconds).

## 5. Verify Installation

```
./scripts/health-check.sh
```

You should see all services marked as “Healthy” or “Running”.

# Running the Application

## Start All Services

```
Development mode (with hot reload)
./scripts/deploy.sh dev

Or manually with docker-compose
docker-compose -f docker-compose.dev.yml up -d
```

## Stop All Services

```
docker-compose down
```

## Restart Specific Service

```
Restart API
docker-compose -f docker-compose.dev.yml restart api

Restart React Admin
docker-compose -f docker-compose.dev.yml restart admin-react
```

## View Logs

```
All services
./scripts/logs.sh all -f

Specific service
./scripts/logs.sh api -f
./scripts/logs.sh admin-react -f

Last 100 lines
./scripts/logs.sh api --tail=100
```

## Development Workflow

---

### Making Code Changes

#### .NET API Backend

```
1. Edit files in backend-dotnet-core/
2. Changes auto-reload (hot reload enabled)
3. View logs
./scripts/logs.sh api -f

If dependencies changed, rebuild:
docker-compose -f docker-compose.dev.yml up -d --build api
```

#### React Admin Panel

```
1. Edit files in admin-panel-react/src/
2. Changes auto-reload in browser
3. View logs
./scripts/logs.sh admin-react -f

If packages changed:
docker-compose -f docker-compose.dev.yml exec admin-react npm install
docker-compose -f docker-compose.dev.yml restart admin-react
```

#### Rust Server Agent

```
1. Edit files in rust-server-agent/src/
2. Changes auto-reload with cargo-watch
3. View logs
./scripts/logs.sh server-agent -f

If Cargo.toml changed:
docker-compose -f docker-compose.dev.yml up -d --build server-agent
```

#### PHP Laravel Admin

```
1. Edit files in admin-panel-php-laravel/
2. Changes reflect immediately
3. View logs
./scripts/logs.sh admin-php -f

If composer.json changed:
docker-compose -f docker-compose.dev.yml exec admin-php composer install
docker-compose -f docker-compose.dev.yml restart admin-php
```

## Git Workflow

```
1. Create feature branch
git checkout -b feature/new-feature

2. Make changes and commit
git add .
git commit -m "Add new feature"

3. Push to remote
git push origin feature/new-feature

4. Create Pull Request on GitHub
```

## Database Management

### Access Database

#### Via pgAdmin (GUI)

1. Open `http://localhost:5050`
2. Login: `admin@caspervpn.local` / `admin123`
3. Add Server:
  - Name: CasperVPN Dev
  - Host: postgres
  - Port: 5432
  - Database: caspervpn\_dev
  - Username: devuser
  - Password: devpass123

#### Via Command Line

```
Connect to PostgreSQL
docker-compose exec postgres psql -U devuser -d caspervpn_dev

List tables
\d

Query data
SELECT * FROM users;

Exit
\q
```

## Database Migrations

### .NET Entity Framework

```
Create migration
docker-compose exec api dotnet ef migrations add MigrationName

Apply migration
docker-compose exec api dotnet ef database update

Rollback
docker-compose exec api dotnet ef database update PreviousMigration
```

## Laravel Migrations

```
Run migrations
docker-compose exec admin-php php artisan migrate

Rollback
docker-compose exec admin-php php artisan migrate:rollback

Fresh migration (reset database)
docker-compose exec admin-php php artisan migrate:fresh
```

## Seed Data

```
Seed database with test data
docker-compose exec admin-php php artisan db:seed

Or specific seeder
docker-compose exec admin-php php artisan db:seed --class=UsersSeeder
```

## Reset Database

```
Stop services
docker-compose down

Remove database volume
docker volume rm caspervpn-postgres-dev-data

Start services (will create fresh database)
./scripts/deploy.sh dev
```

## Testing

### .NET API Tests

```
Run all tests
docker-compose exec api dotnet test

Run specific test
docker-compose exec api dotnet test --filter TestName

With coverage
docker-compose exec api dotnet test /p:CollectCoverage=true
```

### React Tests

```
Run tests
docker-compose exec admin-react npm test

Run tests in watch mode
docker-compose exec admin-react npm test -- --watch

With coverage
docker-compose exec admin-react npm test -- --coverage
```

## Rust Tests

```
Run tests
docker-compose exec server-agent cargo test

Run specific test
docker-compose exec server-agent cargo test test_name

With output
docker-compose exec server-agent cargo test -- --nocapture
```

## PHP Laravel Tests

```
Run tests
docker-compose exec admin-php php artisan test

Run specific test
docker-compose exec admin-php php artisan test --filter TestName

With coverage
docker-compose exec admin-php php artisan test --coverage
```

## Debugging

### .NET API Debugging

#### 1. VS Code:

- Install C# extension
- Add .vscode/launch.json :

```
json
{
 "version": "0.2.0",
 "configurations": [
 {
 "name": ".NET Core Attach",
 "type": "coreclr",
 "request": "attach",
 "processId": "${command:pickRemoteProcess}",
 "pipeTransport": {
 "pipeProgram": "docker",
 "pipeArgs": ["exec", "-i", "caspervpn-api-dev"],
 "debuggerPath": "/vsdbg/vsdbg",
 "pipeCwd": "${workspaceRoot}"
 }
 }
]
}
```

#### 2. Rider:

- Use Docker Run Configuration
- Attach to container

## React Debugging

### 1. Browser DevTools:

- Open Chrome DevTools (F12)
- Go to Sources tab
- Set breakpoints

### 2. VS Code:

- Install Debugger for Chrome extension
- Add launch configuration
- Start debugging (F5)

## Check Logs

```
Real-time logs
./scripts/logs.sh api -f

Error logs only
docker-compose logs api | grep -i error

Last 50 lines
docker-compose logs --tail=50 api
```

## Common Tasks

### Clear Cache

```
Redis cache
docker-compose exec redis redis-cli FLUSHALL

Laravel cache
docker-compose exec admin-php php artisan cache:clear

.NET cache
docker-compose exec api dotnet clean
```

### Update Dependencies

```
.NET packages
docker-compose exec api dotnet restore

NPM packages
docker-compose exec admin-react npm update

Composer packages
docker-compose exec admin-php composer update

Rust dependencies
docker-compose exec server-agent cargo update
```

## Database Backup & Restore

```
Create backup
./scripts/backup.sh

Restore from backup
./scripts/restore.sh backups/YYYYMMDD_HHMMSS.tar.gz
```

## Clean Docker Resources

```
Stop all containers
docker-compose down

Remove unused images
docker image prune -a

Remove unused volumes
docker volume prune

Clean everything
docker system prune -a --volumes
```

## IDE Configuration

### Visual Studio Code

#### Recommended Extensions:

```
Install extensions
code --install-extension ms-dotnettools.csharp
code --install-extension dbaeumer.vscode-eslint
code --install-extension esbenp.prettier-vscode
code --install-extension rust-lang.rust-analyzer
code --install-extension bmewburn.vscode-intelephense-client
code --install-extension ms-azuretools.vscode-docker
```

#### Settings (.vscode/settings.json):

```
{
 "editor.formatOnSave": true,
 "editor.codeActionsOnSave": {
 "source.fixAll.eslint": true
 },
 "[javascript]": {
 "editor.defaultFormatter": "esbenp.prettier-vscode"
 },
 "[csharp]": {
 "editor.defaultFormatter": "ms-dotnettools.csharp"
 }
}
```

## JetBrains IDEs

1. **Open Project:** File → Open → Select `caspervpn` directory
2. **Configure Docker:** Settings → Build → Docker → Add Docker

3. **Run Configuration:** Add Docker Compose configuration
4. **Database:** Add PostgreSQL datasource

## Development Tools

---

### API Testing with Postman

1. Import collection: `docs/postman/CasperVPN.postman_collection.json`
2. Import environment: `docs/postman/Development.postman_environment.json`
3. Test endpoints

### Redis Management

```
Access Redis CLI
docker-compose exec redis redis-cli

Or use Redis Commander
Open http://localhost:8082
```

### Database GUI Tools

- **pgAdmin:** `http://localhost:5050` (included in dev environment)
- **TablePlus:** Connect to `localhost:5432`
- **DBeaver:** Connect to `localhost:5432`

## Troubleshooting

---

### Port Already in Use

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

Kill process or change port in .env
```

### Service Won't Start

```
Check logs
./scripts/logs.sh <service>

Rebuild container
docker-compose -f docker-compose.dev.yml up -d --build <service>

Check Docker resources
docker system df
```

## Database Connection Error

```
Check database is running
docker-compose ps postgres

Test connection
docker-compose exec postgres pg_isready

Restart database
docker-compose restart postgres
```

## Hot Reload Not Working

```
React: Ensure CHOKIDAR_USEPOLLING=true in docker-compose.dev.yml

.NET: Use dotnet watch instead of dotnet run

Rust: Ensure cargo-watch is installed
```

## Best Practices

---

1. **Always work in feature branches**
2. **Commit frequently with clear messages**
3. **Run tests before pushing**
4. **Keep `.env` file secure (never commit)**
5. **Use consistent code formatting**
6. **Write meaningful commit messages**
7. **Update documentation when changing features**
8. **Clean up Docker resources regularly**

## Next Steps

---

- Review [DevOps Overview](#) (./DEVOPS.md)
- Read [Deployment Guide](#) (./DEPLOYMENT.md)
- Check [Monitoring Guide](#) (./MONITORING.md)
- See [Troubleshooting Guide](#) (./TROUBLESHOOTING.md)

## Support

---

Need help?

- Check logs: `./scripts/logs.sh <service>`
- Run health check: `./scripts/health-check.sh`
- Ask team on Slack
- Create GitHub issue

---

**Last Updated:** December 5, 2025

**Version:** 1.0.0