Reward

Release v0.6.6

Janos Miko <info@janosmiko.com>

May 14, 2024

CONTENTS:

Reward is a Swiss Army knife CLI utility for orchestrating Docker based development environments. It makes possible to run multiple local environments simultaneously without port conflicts by utilizing common services proxying requests to the correct environment's containers.

Reward is written in Go, and it supports Linux, macOS and Windows (and WSL2).

CHAPTER

ONE

FEATURES

- Traefik for SSL termination and routing/proxying requests into the correct containers.
- Portainer for quick visibility into what's running inside the local Docker host.
- Dnsmasq to serve DNS responses for **.test* domains eliminating manual editing of */etc/hosts*
- An SSH tunnel for connecting from Sequel Pro or TablePlus into any one of multiple running database containers.
- Reward issued wildcard SSL certificates for running https on all local development domains.
- A common PHPMyAdmin (or optionally Adminer) container for database management.
- An ElasticHQ container for the ability to manage the Elasticsearch Clusters.
- **Full support for Magento 1, Magento 2, PWA Studio, Laravel, Symfony 4, Shopware 6, WordPress and more on all major**
  operating systems
- Ability to override, extend, or setup completely custom environment definitions on a per-project basis.

1.1 Getting Started

1.1.1 Requirements

Installation Summary

Installing Reward is relatively easy. You can just go to the Reward downloads page and get the binary for your platform. You should then extract it to any directory and add that directory to your system's PATH.

But if you prefer, you can use package managers as well. See in the Downloading Reward section of this page. But there are some requirements that you need to meet before you can install Reward.

Common requirements

- **Docker CE 20.10.2** or later
  - Docker for Linux (Reward has been tested on Fedora 31 and Ubuntu 18.04, 20.04).
  - Docker Desktop for Mac 20.10.2 or later.
  - Docker Desktop for Windows 20.10.2 or later.
- **docker-compose version 1.27.4** or later is required (this can be installed via `brew`, `dnf`, or `pip3` as needed). It is currently a part of the Docker Desktop package.

> **Warning:** **Beware of system package managers!** Some operating system distributions include Docker and docker-compose package in their upstream package repos. Please do not install Docker and docker-compose in this manner. Typically these packages are very outdated versions. If you install via your system's package manager, it is very likely that you will experience issues. Please use the official installers on the Docker Install page.

> **Warning:** If you keep getting the **docker api is unreachable** error message, check the FAQ.

Additional requirements (macOS only)

- Mutagen 0.13.1 or later is required for environments leveraging sync sessions on macOS. Reward will attempt to install mutagen via `brew` if not present on macOS.

> **Warning:** **By default Docker Desktop allocates 2GB memory.** This leads to extensive swapping, killed processes and extremely high CPU usage during some Magento actions, like for example running sample-data:deploy and/or installing the application. It is recommended to assign at least 6GB RAM to Docker Desktop prior to deploying any Magento environments on Docker Desktop. This can be corrected via Preferences -> Resources -> Advanced -> Memory. While you are there, it wouldn't hurt to let Docker have the use of a few more vCPUs (keep it at least 4 less than the maximum CPU allocation however to avoid having macOS contend with Docker for use of cores)

Additional requirements (Windows only)

- Mutagen 0.13.1 or later is required for environments leveraging sync sessions on Windows. Reward will attempt to install mutagen to the same path it is installed.
- YogaDNS 1.16 Beta or later is required for using dnsmasq as a local DNS resolver on Windows.

> **Warning:** **On Windows with WSL2 docker can use unlimited memory and CPU.** It is possible and suggested to configure limitations to WSL. You can create a .wslconfig file to your user's home directory with the following content.
>
> *C:\Users\<yourUserName>\.wslconfig*
>
> ```
> [wsl2]
> memory=8GB
> processors=4
> ```
>
> If you configured this you will have to restart WSL with the following PowerShell command:
>
> ```
> Restart-Service LxssManager
> ```
>
> See further instructions here: https://docs.microsoft.com/en-us/windows/wsl/wsl-config#wsl-2-settings

1.1.2 Downloading Reward

Linux

Ubuntu

Step by Step installation on Ubuntu

```
$ curl -fsSLO "https://github.com/rewardenv/reward/releases/latest/download/reward_
↪`uname -s`_`uname -m`.deb"
$ sudo dpkg -i "reward_`uname -s`_`uname -m`.deb"
```

CentOS and Fedora

```
$ yum install -y "https://github.com/rewardenv/reward/releases/latest/download/reward_
↪`uname -s`_`uname -m`.rpm"
```

Binary Download

```
$ curl -fsSLO "https://github.com/rewardenv/reward/releases/latest/download/reward_
↪`uname -s`_`uname -m`.tar.gz"
$ tar -zxvf "reward_`uname -s`_`uname -m`.tar.gz" -C /usr/local/bin/
$ rm -f "reward_`uname -s`_`uname -m`.tar.gz"
$ chmod +x /usr/local/bin/reward
```

---

macOS

You can install reward using Homebrew or by downloading the binary itself and putting it to PATH.

Using Homebrew

```
$ brew install rewardenv/tap/reward
```

Binary download

```
$ curl -fsSLO "https://github.com/rewardenv/reward/releases/latest/download/reward_
↪`uname -s`_`uname -m`.tar.gz"
$ tar -zxvf "reward_`uname -s`_`uname -m`.tar.gz" -C /usr/local/bin/
$ rm -f "reward_`uname -s`_`uname -m`.tar.gz"
$ chmod +x /usr/local/bin/reward
```

Windows

Download Reward from this link and extract to any folder like `C:\bin`. Please make sure that folder is in you PATH environment variable.

You can find a nice guide here about how to configure PATH in Windows.

Updating Reward

When Reward is already installed on your system, you can do a self-update running `reward self-update` command.

If you installed it using a package manager on linux, you will have to run it as superuser with `sudo reward self-update`.

Next Steps

You will have to run `reward install` to initialize Reward. See more in the Setting Up

1.1.3 Setting up Reward

Install Reward

Before you first run Reward, you will have to install Reward configurations.

```
$ reward install
```

This is going to do the following:

- Create a Self Signed Root CA Certificate and install it to you operating system's Root CA Trust. To do so Reward will ask for your sudo / administrator permission.
- Configure your Operating System's DNS resolver to use Reward's dnsmasq service to resolve *.test domains (macOS and Linux only).
- Create an SSH Tunnel Key and configure SSH (/etc/ssh/ssh_config) to use this key if you want to utilize Reward's tunnel (macOS and Linux only).

Provision Reward's Global Services

After you installed Reward's basic settings you'll be able to provision the global services such as Traefik, Portainer and so on.

To do that, run the following command:

```
$ reward svc up
```

For more information check the *Global Services* guide.

Automatic DNS Resolution

Linux

On Linux environments, Reward tries to configure your NetworkManager and systemd-resolved to use the local resolver. If it's not working, you will have to configure your DNS to resolve `*.test` to `127.0.0.1` or use `/etc/hosts` entries.

After Reward is installed, probably you will have to restart your NetworkManager (or reboot your system).

For further information, see the Automatic DNS Resolution guide.

macOS

This configuration is automatic via the BSD per-TLD resolver configuration found at `/etc/resolver/test`.

Windows

We suggest to use YogaDNS which allows you to create per domain rules for DNS resolution. With YogaDNS you can configure your OS to ask dnsmasq for all `*.test` domain and use your default Name Server for the rest.

For more information see the configuration page for Automatic DNS Resolution

---

Trusted CA Root Certificate

In order to sign SSL certificates that may be trusted by a developer workstation, Reward uses a CA root certificate with CN equal to `Reward Proxy Local CA (<hostname>)` where `<hostname>` is the hostname of the machine the certificate was generated on at the time Reward was first installed. The CA root can be found at `~/.reward/ssl/rootca/certs/ca.cert.pem`.

Linux

On Ubuntu/Debian this CA root is copied into `/usr/local/share/ca-certificates` and on Fedora/CentOS (Enterprise Linux) it is copied into `/etc/pki/ca-trust/source/anchors` and then the trust bundle is updated appropriately. For new systems, this typically is all that is needed for the CA root to be trusted on the default Firefox browser, but it may not be trusted by Chrome or Firefox automatically should the browsers have already been launched prior to the installation of Reward (browsers on Linux may and do cache CA bundles)

macOS

On macOS this root CA certificate is automatically added to a users trust settings as can be seen by searching for ' Reward Proxy Local CA' in the Keychain application. This should result in the certificates signed by Reward being trusted by Safari and Chrome automatically. If you use Firefox, you will need to add this CA root to trust settings specific to the Firefox browser per the below.

Windows

On Windows this root CA certificate is automatically added to the users trust settings as can be seen by searching for ‘ Reward Proxy Local CA’ in the Management Console. This should result in the certificates signed by Reward being trusted by Edge, Chrome and Firefox automatically.

Note: If you are using **Firefox** and it warns you the SSL certificate is invalid/untrusted, go to Preferences -> Privacy & Security -> View Certificates (bottom of page) -> Authorities -> Import and select `~/.reward/ssl/rootca/certs/ca.cert.pem` for import and make sure you select **‘Trust this CA to identify websites’**. Then reload the page.

If you are using **Chrome** on **Linux** and it warns you the SSL certificate is invalid/untrusted, go to Chrome Settings -> Privacy And Security -> Manage Certificates (see more) -> Authorities -> Import and select `~/.reward/ssl/rootca/certs/ca.cert.pem` for import and make sure you select **‘Trust this certificate for identifying websites’**. Then reload the page.

1.1.4 Step-by-step installation on Ubuntu

Uninstall old versions of Docker

Older versions of Docker were called docker, docker.io, or docker-engine. If these are installed, uninstall them:

```
sudo apt-get remove docker docker-engine docker.io containerd runc -y
```

Install using the repository

Set up the repository

- Update the apt package index and install packages to allow apt to use a repository over HTTPS:

```
sudo apt-get update
```

```
sudo apt-get install -y \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

- Add Docker’s official GPG key:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/
↪share/keyrings/docker-archive-keyring.gpg
```

- Use the following command to set up the stable repository.

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-
↪keyring.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Install Docker Engine

- Update the apt package index, and install the latest version of Docker Engine and containerd

```
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io
```

- Verify that Docker Engine is installed correctly by running the hello-world image.

```
sudo docker run hello-world
```

- Add your user to docker group to be able to use docker without sudo

```
sudo usermod -aG docker $USER
```

Now you have to **log out and log back** in to apply the user group change.

Install Docker Compose

Prerequisites

Docker Compose relies on Docker Engine for any meaningful work, so make sure you have Docker Engine installed. (See the previous step.)

- There are multiple ways to install `docker-compose`, we are going to use python3-pip, so make sure it's installed on your system.

```
sudo apt-get install -y python3-pip
```

- Run this command to install the current stable release of Docker Compose using pip:

```
sudo pip install docker-compose
```

- Test the installation. It should be version >= `1.26`.

```
docker-compose --version
```

Installing Reward

- Download the latest package and install it with dpkg.

```
curl -fsSLO "https://github.com/rewardenv/reward/releases/latest/download/reward_`uname -
↪s`_`uname -m`.deb"
sudo dpkg -i "reward_`uname -s`_`uname -m`.deb"
```

```
reward install
```

Warning: If you face the error message *docker api is unreachable*, please make sure your user is in the docker group. Don't forget to relog (log out and log back in).

- Verify the installation

```
reward --version
```

1.2 Global Services

After running `reward svc up` for the first time following installation, the following URLs can be used to interact with the UIs for services Reward runs globally:

- https://traefik.reward.test/
- https://portainer.reward.test/
- https://dnsmasq.reward.test/
- https://mailbox.reward.test/ or https://mb.reward.test/
- https://phpmyadmin.reward.test/ or https://pma.reward.test/
- https://elastichq.reward.test/
- optional: https://adminer.reward.test/

1.2.1 Customizable Settings

When spinning up global services via `docker-compose` Reward uses `~/.reward` as the project directory and `~/.reward.yml` or `~/.reward/.env` to function for overriding variables in the `docker-compose` configuration used to deploy these services.

The following options are available (with default values indicated):

- `TRAEFIK_LISTEN=127.0.0.1` may be set to `0.0.0.0` for example to have Traefik accept connections from other devices on the local network.
- `REWARD_RESTART_POLICY=always` may be set to `no` to prevent Docker from restarting these service containers or any other valid restart policy value.
- `REWARD_SERVICE_DOMAIN=reward.test` may be set to a domain of your choosing if so desired. Please note that this will not currently change network settings or alter `dnsmasq` configuration. Any TLD other than `test` will require DNS resolution be manually configured.

Warning: Setting `TRAEFIK_LISTEN=0.0.0.0` can be quite useful in some cases, but be aware that causing Traefik to listen for requests publicly poses a security risk when on public WiFi or networks otherwise outside of your control.

After changing settings in `~/.reward.yml` or `~/.reward/.env`, please run `reward svc up` to apply.

1.3 Environments

1.3.1 Environment Types

Reward currently supports 8 environment types.

- Magento 1
- Magento 2
- PWA Studio (for Magento 2)
- Laravel
- Symfony
- Shopware
- WordPress
- Generic PHP
- Local

  These types are passed to `env-init` when configuring a project for local development for the first time. This list of environment types can also be seen by running `reward env-init --help` on your command line. The `docker-compose` configuration used to assemble each environment type can be found in the templates directory on GitHub.

Magento 1

The `magento1` environment type supports development of Magento 1 projects, launching containers including:

- Nginx
- PHP-FPM (5.6 or 7.0+)
- MariaDB
- Redis

Files are currently mounted using a delegated mount on macOS/Windows and natively on Linux.

Magento 2

The `magento2` environment type provides necessary containerized services for running Magento 2 in a local development context including:

- Nginx
- Varnish
- PHP-FPM (7.0+)
- MariaDB
- Elasticsearch
- RabbitMQ
- Redis

In order to achieve a well performing experience on macOS and Windows, files in the webroot are synced into the container using a Mutagen sync session except `pub/media` which remains mounted using a delegated mount.

PWA Studio

The `pwa-studio` environment type provides necessary containerized services for running PWA in a local development context including:

- NodeJS (with yarn)

Laravel

The `laravel` environment type supports development of Laravel projects, launching containers including:

- Nginx
- PHP-FPM
- MariaDB
- Redis

Files are currently mounted using a delegated mount on macOS/Windows and natively on Linux.

Symfony

The `symfony` environment type supports development of Symfony 4+ projects, launching containers including:

- Nginx
- PHP-FPM
- MariaDB
- Redis
- RabbitMQ (disabled by default)
- Varnish (disabled by default)
- Elasticsearch (disabled by default)

Files are currently mounted using a delegated mount on macOS/Windows and natively on Linux.

Shopware

The `shopware` environment type supports development of Shopware 6 projects, launching containers including:

- Nginx
- PHP-FPM
- MariaDB
- Redis
- RabbitMQ (disabled by default)
- Varnish (disabled by default)
- Elasticsearch (disabled by default)

In order to achieve a well performing experience on macOS and Windows, files in the webroot are synced into the container using a Mutagen sync session except `public/media` which remains mounted using a delegated mount.

WordPress

The `wordpress` environment type supports development of WordPress 5 projects, launching containers including:

- Nginx
- PHP-FPM
- MariaDB
- Redis (disabled by default)

In order to achieve a well performing experience on macOS and Windows, files in the webroot are synced into the container using a Mutagen sync session except `wp-content/uploads` which remains mounted using a delegated mount.

Generic PHP

The `generic-php` environment type contains nginx, php-fpm, php-debug, database (and an optional redis) containers.

Using this type, you will get a more generic development environment, with just serving the files from the current directory.

It is useful for any other PHP frameworks and raw PHP development.

Local

The `local` environment type does nothing more than declare the `docker-compose` version and label the project network so Reward will recognize it as belonging to an environment orchestrated by Reward.

When this type is used, a `.reward/reward-env.yml` may be placed in the root directory of the project workspace to define the desired containers, volumes, etc needed for the project. An example of a `local` environment type being used can be found in the Initializing a Custom Node Environment in a Subdomain or in m2demo project.

Similar to the other environment type's base definitions, Reward supports a `reward-env.darwin.yml`, `reward-env.linux.yml` and `reward-env.windows.yml`

Commonalities

In addition to the above, each environment type (except the `local` type) come with PHP setup to use `msmtp` to ensure outbound email does not inadvertently leave your network and to support simpler testing of email functionality. Mailbox may be accessed by navigating to https://mailbox.reward.test/ in a browser.

Where PHP is specified in the above list, there should be two `fpm` containers, `php-fpm` and `php-debug` in order to provide Xdebug support. Use of Xdebug is enabled by setting the `XDEBUG_SESSION` cookie in your browser to direct the request to the `php-debug` container. Shell sessions opened in the debug container via `reward debug` will also connect PHP processes for commands on the CLI to Xdebug.

The configuration of each environment leverages a `base` configuration YAML file, and optionally a `darwin` and `linux` file to add to `base` configuration anything which may be specific to a given host architecture (this is, for example, how the `magento2` environment type works seamlessly on macOS with Mutagen sync sessions while using native filesystem mounts on Linux hosts).

1.3.2 Initializing Laravel

Initializing an Empty Laravel Project

1. Create an empty directory and a Reward Laravel environment

```
$ mkdir ~/Sites/your-awesome-laravel-project
$ reward env-init your-awesome-laravel-project --environment-type=laravel
```

2. Sign a new certificate for your dev domain

```
$ reward sign-certificate your-awesome-laravel-project.test
```

3. Bring up the Reward environment

```
$ reward env up
```

4. Create the laravel project in the php container

```
$ reward shell

$ composer create-project --no-install --no-scripts --prefer-dist \
    laravel/laravel /tmp/laravel-tmp
$ rsync -au --remove-source-files /tmp/laravel-tmp/ /var/www/html/
```

5. Install the composer packages and create an app key

```
$ reward shell

$ composer install

$ php artisan key:generate --show

# Add the previously generated key to your .env file with your favourite editor
# It should be added using the following format
APP_KEY=base64:yourkey

# Import the new .env content to the runtime environment variables
$ source .env

# Generate your config cache
$ php artisan config:cache
```

Note: Now you can reach the project on the following url:

https://your-awesome-laravel-project.test

Initializing a Laravel Backpack Demo Project

1. Clone the code and initialize a Laravel Reward environment

```
$ git clone https://github.com/Laravel-Backpack/demo.git ~/Sites/demo
$ cd ~/Sites/demo
$ reward env-init demo --environment-type=laravel
```

2. Sign a new certificate for your dev domain

```
$ reward sign-certificate demo.test
```

3. Bring up the Reward environment

```
$ reward env up
```

4. Install the composer packages and initialize the database

```
$ reward shell

$ composer install
$ php artisan key:generate --show

# Add the previously generated key to your .env file with your favourite editor
# It should be added using the following format
APP_KEY=base64:yourkey

# Import the new .env content to the runtime environment variables
$ source .env

$ php artisan migrate
$ php artisan db:seed
```

Note: Now you can reach the project on the following url:

https://demo.test

The default admin credentials are the following:

```
user: admin@example.com
pass: admin
```

1.3.3 Initializing Magento 1

Importing a Magento 1 Project and initializing with bootstrap command

1. Clone your project and initialize Reward.

```
$ git clone git://github.com/your-user/your-awesome-m1-project.git ~/Sites/your-
↪awesome-m1-project
$ cd ~/Sites/your-awesome-m1-project
$ reward env-init your-awesome-m2-project --environment-type=magento1
```

2. Before running the bootstrap command, you should import the Magento database to the DB Container. To do so, first start the DB container:

```
$ reward env up -- db
```

3. Import the database.

```
$ reward db import < /path/to/db-dump-for-magento1.sql
```

4. When the import is done, you can run the bootstrap.

```
$ reward bootstrap
```

1.3.4 Initializing Magento 2

Empty Magento 2 Project with bootstrap command

It's pretty easy to bootstrap a Magento 2 project using Reward.

1. Create a new environment in an empty directory:

```
$ mkdir ~/Sites/your-awesome-m2-project
$ cd ~/Sites/your-awesome-m2-project
$ reward env-init your-awesome-m2-project
```

2. If this is the first time you install a magento project, you have to configure your composer keys:

```
$ reward env up -- php-fpm
$ reward shell
$ composer config -a -g http-basic.repo.magento.com MAGENTO_PUBLIC_KEY MAGENTO_
↪PRIVATE_KEY
```

3. Provision the environment using Reward's bootstrap command:

```
$ reward bootstrap
```

This is going to create a new Magento 2 installation using Composer (if your Composer Magento Repo Key is not set it will ask for it). After the vendor installation it's going to install Magento (configure the env.php), configure the local domains and configure an admin user.

Note: Bootstrap Options:

- `--crypt-key`: specify the magento encryption key
- `--db-prefix`: specify db prefix for magento
- `--disable-tfa`: disable magento two factor auth
- `--full`: include sampledata and reindexing
- `--magento-mode`: specify magento run mode (developer, default, production)
- `--magento-type`: specify the magento type (community or enterprise)
- `--magento-version`: magento version
- `--reset-admin-url`: reset the admin url after the installation

- `--skip-composer-install`: bootstrap without composer install (if it's already installed)
- `--with-sampledata`: install magento with sample data

Importing a Magento 2 Project and initializing with bootstrap command

1. Clone your project and initialize Reward.

```
$ git clone git://github.com/your-user/your-awesome-m2-project.git ~/Sites/your-
↪awesome-m2-project
$ cd ~/Sites/your-awesome-m2-project
$ reward env-init your-awesome-m2-project
```

2. Before running the bootstrap command, you should import the Magento database to the DB Container. To do so, first start the DB container:

```
$ reward env up -- db
```

3. Import the database.

```
$ reward db import < /path/to/db-dump.sql
```

4. If this is the first time you install a magento project, you have to configure your composer keys:

```
$ reward env up -- php-fpm
$ reward shell
$ composer config -a -g http-basic.repo.magento.com MAGENTO_PUBLIC_KEY MAGENTO_
↪PRIVATE_KEY
```

5. When the import is done, you can run the bootstrap.

```
$ reward bootstrap
```

Note: If you already installed your composer component, it's also possible to skip the composer install steps:

```
$ reward bootstrap --skip-composer-install
```

Initializing A Magento 2 Environment Manually

The below example demonstrates the from-scratch setup of the Magento 2 application for local development. A similar process can easily be used to configure an environment of any other type. This assumes that Reward has been previously started via `reward svc up` as part of the installation procedure.

1. Create a new directory on your host machine at the location of your choice and then jump into the new directory to get started:

```
$ mkdir -p ~/Sites/your-awesome-m2-project
$ cd ~/Sites/your-awesome-m2-project
```

2. From the root of your new project directory, run `env-init` to create the `.env` file with configuration needed for Reward and Docker to work with the project.

```
$ reward env-init your-awesome-m2-project --environment-type magento2
```

The result of this command is a `.env` file in the project root (tip: commit this to your VCS to share the configuration with other team members) having the following contents:

```
REWARD_ENV_NAME=your-awesome-m2-project
REWARD_ENV_TYPE=magento2
REWARD_WEB_ROOT=/

TRAEFIK_DOMAIN=your-awesome-m2-project.test
TRAEFIK_SUBDOMAIN=
TRAEFIK_EXTRA_HOSTS=

REWARD_DB=true
REWARD_ELASTICSEARCH=false
REWARD_OPENSEARCH=true
REWARD_OPENSEARCH_DASHBOARDS=false
REWARD_VARNISH=true
REWARD_RABBITMQ=true
REWARD_REDIS=true

REWARD_SYNC_IGNORE=

ELASTICSEARCH_VERSION=7.12
OPENSEARCH_VERSION=1.2
MARIADB_VERSION=10.4
NODE_VERSION=16
PHP_VERSION=7.3
RABBITMQ_VERSION=3.8
REDIS_VERSION=6.0
VARNISH_VERSION=6.5

REWARD_ALLURE=false
REWARD_SELENIUM=false
REWARD_SELENIUM_DEBUG=false
REWARD_BLACKFIRE=false
REWARD_SPLIT_SALES=false
REWARD_SPLIT_CHECKOUT=false
REWARD_TEST_DB=false
REWARD_MAGEPACK=false

BLACKFIRE_CLIENT_ID=
BLACKFIRE_CLIENT_TOKEN=
BLACKFIRE_SERVER_ID=
BLACKFIRE_SERVER_TOKEN=
```

3. Sign an SSL certificate for use with the project (the input here should match the value of `TRAEFIK_DOMAIN` in the above `.env` example file):

```
$ reward sign-certificate your-awesome-m2-project.test
```

4. Next you'll want to start the project environment:

```
$ reward env up
```

> **Warning:** If you encounter an error about `Mounts denied`, follow the instructions in the error message and run `reward env up -d` again.

5. Drop into a shell within the project environment. Commands following this step in the setup procedure will be run from within the `php-fpm` docker container this launches you into:

```
$ reward shell
```

6. Configure global Magento Marketplace credentials

```
$ composer global config http-basic.repo.magento.com <username> <password>
```

Note: To locate your authentication keys for Magento 2 repository, reference DevDocs.

If you have previously configured global credentials, you may skip this step, as `~/.composer/` is mounted into the container from the host machine in order to share composer cache between projects, and also shares the global `auth.json` from the host machine.

---

7. Initialize project source files using composer create-project and then move them into place:

```
META_PACKAGE=magento/project-community-edition META_VERSION=2.4.x

$ composer create-project --no-install --repository-url=https://repo.magento.com/ \
    "${META_PACKAGE}" /tmp/magento-tmp "${META_VERSION}"

$ rsync -au --remove-source-files /tmp/magento-tmp/ /var/www/html/

$ composer install
```

8. Install the application, and you should be all set:

```
# Install Application
bin/magento setup:install \
    --backend-frontname=backend \
    --amqp-host=rabbitmq \
    --amqp-port=5672 \
    --amqp-user=guest \
    --amqp-password=guest \
    --db-host=db \
    --db-name=magento \
    --db-user=magento \
    --db-password=magento \
    --search-engine=elasticsearch7 \
    --elasticsearch-host=elasticsearch \
    --elasticsearch-port=9200 \
    --elasticsearch-index-prefix=magento2 \
    --elasticsearch-enable-auth=0 \
    --elasticsearch-timeout=15 \
    --http-cache-hosts=varnish:80 \
```

(continues on next page)

(continued from previous page)

```
    --session-save=redis \
    --session-save-redis-host=redis \
    --session-save-redis-port=6379 \
    --session-save-redis-db=2 \
    --session-save-redis-max-concurrency=20 \
    --cache-backend=redis \
    --cache-backend-redis-server=redis \
    --cache-backend-redis-db=0 \
    --cache-backend-redis-port=6379 \
    --page-cache=redis \
    --page-cache-redis-server=redis \
    --page-cache-redis-db=1 \
    --page-cache-redis-port=6379

# Configure Application
bin/magento config:set --lock-env web/unsecure/base_url \
    "https://${TRAEFIK_SUBDOMAIN:+$TRAEFIK_SUBDOMAIN.}${TRAEFIK_DOMAIN}/"

bin/magento config:set --lock-env web/secure/base_url \
    "https://${TRAEFIK_SUBDOMAIN:+$TRAEFIK_SUBDOMAIN.}${TRAEFIK_DOMAIN}/"

bin/magento config:set --lock-env web/secure/offloader_header X-Forwarded-Proto

bin/magento config:set --lock-env web/secure/use_in_frontend 1
bin/magento config:set --lock-env web/secure/use_in_adminhtml 1
bin/magento config:set --lock-env web/seo/use_rewrites 1

bin/magento config:set --lock-env system/full_page_cache/caching_application 2
bin/magento config:set --lock-env system/full_page_cache/ttl 604800

bin/magento config:set --lock-env catalog/search/enable_eav_indexer 1

bin/magento config:set --lock-env dev/static/sign 0

bin/magento deploy:mode:set -s developer
bin/magento cache:disable block_html full_page

bin/magento indexer:reindex
bin/magento cache:flush
```

Note: Prior to Magento `2.4.x` it was not required to enter search-engine and elasticsearch configuration during installation and these params to `setup:install` are not supported by Magento `2.3.x`. These should be omitted on older versions where not supported and Elasticsearch configured via `config:set` instead:

```
bin/magento config:set --lock-env catalog/search/engine elasticsearch7
bin/magento config:set --lock-env catalog/search/elasticsearch7_server_hostname␣
↪elasticsearch
bin/magento config:set --lock-env catalog/search/elasticsearch7_server_port 9200
bin/magento config:set --lock-env catalog/search/elasticsearch7_index_prefix␣
↪magento2
bin/magento config:set --lock-env catalog/search/elasticsearch7_enable_auth 0
```

(continues on next page)

(continued from previous page)

```
bin/magento config:set --lock-env catalog/search/elasticsearch7_server_timeout 15
```

9. Generate an admin user and configure 2FA for OTP

```
# Generate localadmin user
ADMIN_PASS="$(pwgen -n1 16)"
ADMIN_USER=localadmin

bin/magento admin:user:create \
    --admin-password="${ADMIN_PASS}" \
    --admin-user="${ADMIN_USER}" \
    --admin-firstname="Local" \
    --admin-lastname="Admin" \
    --admin-email="${ADMIN_USER}@example.com"

printf "u: %s\np: %s\n" "${ADMIN_USER}" "${ADMIN_PASS}"

# Configure 2FA provider
OTPAUTH_QRI=
TFA_SECRET=$(python -c "import base64; print base64.b32encode('$(pwgen -A1 128)')"␣
↪| sed 's/=*$//')
OTPAUTH_URL=$(printf "otpauth://totp/%s%%3Alocaladmin%%40example.com?issuer=%s&
↪secret=%s" \
    "${TRAEFIK_SUBDOMAIN}.${TRAEFIK_DOMAIN}" "${TRAEFIK_SUBDOMAIN}.${TRAEFIK_DOMAIN}
↪" "${TFA_SECRET}"
)

bin/magento config:set --lock-env twofactorauth/general/force_providers google
bin/magento security:tfa:google:set-secret "${ADMIN_USER}" "${TFA_SECRET}"

printf "%s\n\n" "${OTPAUTH_URL}"
printf "2FA Authenticator Codes:\n%s\n" "$(oathtool -s 30 -w 10 --totp --base32 "$
↪{TFA_SECRET}")"

segno "${OTPAUTH_URL}" -s 4 -o "pub/media/${ADMIN_USER}-totp-qr.png"
printf "%s\n\n" "https://${TRAEFIK_SUBDOMAIN}.${TRAEFIK_DOMAIN}/media/${ADMIN_USER}-
↪totp-qr.png?t=$(date +%s)"
```

Note: Use of 2FA is mandatory on Magento `2.4.x` and setup of 2FA should be skipped when installing `2.3.x` or earlier. Where 2FA is setup manually via UI upon login rather than using the CLI commands above, the 2FA configuration email may be retrieved from the Mailbox service.

10. Launch the application in your browser:

    - https://your-awesome-m2-project.test/
    - https://your-awesome-m2-project.test/backend/
    - https://rabbitmq.your-awesome-m2-project.test/
    - https://elasticsearch.your-awesome-m2-project.test/

Note: To completely destroy the `your-awesome-m2-project` environment we just created, run `reward env down -v` to tear down the project's Docker containers, volumes, etc.

1.3.5 Initializing PWA Studio

Prerequisites in Magento

From PWA Studio 12.1 you have to install PWA Studio metapackage in your Magento instance.

If you run Magento as a Reward Environment, run the following commands in the PHP container:

```
composer require magento/pwa

# If you want to install sample data
composer require magento/venia-sample-data

php bin/magento setup:upgrade
php bin/magento setup:di:compile
php bin/magento setup:static-content:deploy -f
php bin/magento indexer:reindex
php bin/magento cache:clean
```

Running Default PWA Studio

1. Clone your project and initialize Reward.

   ```
   $ git clone https://github.com/magento/pwa-studio.git ~/Sites/pwa-studio
   $ cd ~/Sites/pwa-studio
   $ reward env-init pwa-studio --environment-type=pwa-studio
   ```

2. Sign a certificate for your project

   ```
   $ reward sign-certificate pwa-studio.test
   ```

3. Fill up the `.env` file with samples and change some settings

   ```
   $ cat docker/.env.docker.dev >> .env
   ```

   ```
   NODE_VERSION=16
   DEV_SERVER_HOST=0.0.0.0
   DEV_SERVER_PORT=8000
   MAGENTO_BACKEND_EDITION=MOS
   ```

4. Update `package.json` and add these to the `scripts` part.

   Note: it seems like PWA Studio 12.x ignores the `DEV_SERVER_PORT` variable, so we override it from the command line.

   ```
   "watch": "yarn watch:venia --disable-host-check --public pwa-studio.test --port 8000
   ↪",
   "start": "yarn stage:venia --disable-host-check --public pwa-studio.test --port 8000
   ```

(continues on next page)

(continued from previous page)

```
↪"

# you can use this script if you are familiar with jq
DOMAIN="pwa-studio.test"
cat package.json | jq --arg domain "$DOMAIN" -Mr '. * {scripts:{watch: ("yarn␣
↪watch:venia --public " + $domain + " --disable-host-check --port 8000"), start: (
↪"yarn stage:venia --public " + $domain + " --disable-host-check --port 8000")}}'␣
↪| tee package.json
```

Note: We have to add both –disable-host-check (to skip host header verification) and –public (to let webpack dev server know it is behind a proxy and it shouldn't add custom port to it's callback URLs).

https://webpack.js.org/configuration/dev-server/#devserverpublic

---

5. Bring up the environment

```
$ reward env up
```

6. Install its dependencies

```
$ reward shell
$ yarn install
```

7. Restart the PWA container

```
$ reward env restart
```

8. Optional: if you'd like to run the project in Developer/Production mode, add the following line to your `.env` file

```
# Developer Mode (default)
DOCKER_START_COMMAND="yarn watch"

# Production Mode
DOCKER_START_COMMAND="yarn start"
```

Reach a Reward Magento backend environment

If your PWA's Magento backend is also running on your computer as a Reward environment, you will have to configure the PWA container to resolve the Magento DNS to the Reward Traefik container.

To do so add a space separated list of domains to the `TRAEFIK_EXTRA_HOSTS` variable in the .env file.

- `TRAEFIK_EXTRA_HOSTS="otherproject.test thirdproject.test"`

1.3.6 Initializing Shopware

Initializing an Empty Shopware Development Project

1. Clone the code and initialize a Reward Shopware environment

```
git clone https://github.com/shopware/development.git -b v6.4.11.1 ~/Sites/your-
↪awesome-shopware-project/webroot
cd ~/Sites/your-awesome-shopware-project
reward env-init your-awesome-shopware-project --environment-type=shopware
```

> **Note:** In this example the shopware code will live in the $PROJECT/webroot directory. If you'd like to use a different directory, change *REWARD_WEB_ROOT* environment variable in *.env* file.

2. Sign a new certificate for your dev domain

```
reward sign-certificate your-awesome-shopware-project.test
```

3. Bring up the Reward environment

```
reward env up
```

4. Install Shopware

   If you use different domain, make sure to **update the `APP_URL`** in the `.psh.yaml.override` file.

```
reward shell

echo $'const:\n  APP_ENV: "dev"\n  APP_URL: "https://your-awesome-shopware-project.
↪test"\n  DB_HOST: "db"\n  DB_NAME: "shopware"\n  DB_USER: "app"\n  DB_PASSWORD:
↪"app"' > .psh.yaml.override

# Windows only: give run permissions for the necessary files
chmod +x psh.phar bin/console bin/setup

./psh.phar install
```

Note: Now you can reach the project on the following url:

https://your-awesome-shopware-project.test

Or the admin dashboard on https://your-awesome-shopware-project.test/admin

user: admin password: shopware

Initializing an Empty Shopware Production Project

1. Clone the code and initialize a Reward Shopware Production environment

```
git clone https://github.com/shopware/production.git -b v6.4.18.0 ~/Sites/your-
↪awesome-shopware-project/webroot
cd ~/Sites/your-awesome-shopware-project
reward env-init your-awesome-shopware-project --environment-type=shopware
```

Note: In this example the shopware code will live in the $PROJECT/webroot directory. If you'd like to use a different directory, change *REWARD_WEB_ROOT* environment variable in *.env* file.

---

2. Sign a new certificate for your dev domain

```
reward sign-certificate your-awesome-shopware-project.test
```

3. Make some fixes in the .env file

```
# Enable opensearch
# REWARD_OPENSEARCH=true
sed -i.bak "s/^REWARD_OPENSEARCH=.*/REWARD_OPENSEARCH=true/g" .env

# Change node version
# NODE_VERSION=16
sed -i.bak "s/^NODE_VERSION=.*/NODE_VERSION=16/g" .env
```

4. Bring up the Reward environment

```
reward env up
```

5. Install Shopware

```
reward shell

# Issue with composer version 2.5
# https://github.com/shopware/production/issues/168
# Rollback to a previous composer version
sudo composer self-update 2.4.4

composer install --no-interaction

# Configure Shopware
bin/console system:setup --no-interaction --app-env dev --app-url https://your-
↪awesome-shopware-project.test --database-url mysql://app:app@db:3306/shopware --
↪es-enabled=1 --es-hosts=opensearch:9200 --es-indexing-enabled=1 --cdn-
↪strategy=physical_filename --mailer-url=native://default

# Install Shopware
bin/console system:install --no-interaction --drop-database --create-database --
↪basic-setup

# Dump plugin and theme settings
export CI=1
```

(continues on next page)

(continued from previous page)

```
bin/console bundle:dump
bin/console theme:dump

# Build storefront
export PUPPETEER_SKIP_CHROMIUM_DOWNLOAD=1
bin/build.sh

# Do the migrations and install assets
bin/console system:update:finish --no-interaction
```

Note: Now you can reach the project on the following url:

https://your-awesome-shopware-project.test

Or the admin dashboard on https://your-awesome-shopware-project.test/admin

user: admin password: shopware

1.3.7 Initializing Shopware PWA

Running Shopware PWA

This guide is based on the official documentation of Shopware PWA.

(0.) Install SwagShopwarePwa Plugin

1. Download a plugin packed in a zip file from github: master version
2. Log in to the admin panel at https://your-awesome-shopware-project.test/admin
3. Go to Setting > System > Plugins and click Upload plugin button.
4. When the plugin is uploaded - just install and activate it. That's all. Shopware 6 is shopware-pwa ready now!

Or to do it programmatically run the following commands inside your Shopware Reward Shell:

```
wget https://github.com/elkmod/SwagShopwarePwa/archive/master.zip -O ~/swpwa.zip
php bin/console plugin:zip-import ~/swpwa.zip
```

---

1. Initialize Reward.

   ```
   $ mkdir ~/Sites/shopware-pwa
   $ cd ~/Sites/shopware-pwa
   $ reward env-init shopware-pwa --environment-type=shopware-pwa
   ```

2. Sign a certificate for your project

   ```
   $ reward sign-certificate shopware-pwa.test
   ```

3. Bring up the environment

   ```
   $ reward env up
   ```

4. Create a regular PWA Project

```
$ reward shell
$ npx @shopware-pwa/cli@canary init
```

5. Configure backend in `shopware-pwa.config.js`
6. Update `package.json` and add these to the `scripts` part

```
"start": "yarn && yarn build --ci && node scripts/init.js --disable-host-check --
↪public shopware-pwa.test"

# you can use this script if you are familiar with jq
DOMAIN="shopware-pwa.test"
cat package.json | jq --arg domain "$DOMAIN" -Mr '. * {scripts:{start: ("yarn &&␣
↪yarn build --ci && node scripts/init.js --public " + $domain + " --disable-host-
↪check")}}' | tee package.json
```

Note: We have to add both –disable-host-check (to skip host header verification) and –public (to let webpack dev server know it is behind a proxy and it shouldn't add custom port to it's callback URLs).

https://webpack.js.org/configuration/dev-server/#devserverpublic

---

7. Install dependencies

```
$ reward shell
$ yarn
```

8. Restart the PWA container

```
$ reward env restart
```

9. Optional: if you'd like to run the project in Developer/Production mode, add the following line to your `.env` file

```
# Developer Mode (default)
DOCKER_START_COMMAND="yarn dev"

# Production Mode
DOCKER_START_COMMAND="yarn start"
```

1.3.8 Initializing Symfony

Initializing an Empty Symfony Project

1. Create an empty directory and a Reward Symfony environment

```
$ mkdir ~/Sites/your-awesome-symfony-project
$ reward env-init your-awesome-symfony-project --environment-type=symfony
```

2. Sign a new certificate for your dev domain

```
$ reward sign-certificate your-awesome-symfony-project.test
```

3. Bring up the Reward environment

```
$ reward env up
```

4. Create the symfony project in the php container

```
$ reward shell

$ composer create-project --no-install --no-interaction \
    symfony/website-skeleton /tmp/symfony-tmp
$ rsync -au --remove-source-files /tmp/symfony-tmp/ /var/www/html/
```

5. Install the composer packages

```
$ reward shell

$ composer install
```

Note: Now you can reach the project on the following url:

https://your-awesome-symfony-project.test

1.3.9 Initializing WordPress

Empty WordPress Project with bootstrap command

It's pretty easy to bootstrap a WordPress project using Reward.

1. Create a new environment in an empty directory:

```
$ mkdir ~/Sites/your-awesome-wordpress-project
$ cd ~/Sites/your-awesome-wordpress-project
$ reward env-init your-awesome-wordpress-project --environment-type=wordpress
```

2. Provision the environment using Reward's bootstrap command:

```
$ reward bootstrap
```

This is going to create a new WordPress installation by downloading WordPress and configuring wp-config.php.

It is possible to change the DB prefix with the following command.

```
$ reward bootstrap --db-prefix=<somestring>
```

Importing a WordPress Project and initializing with bootstrap command

1. Clone your project and initialize Reward.

```
$ git clone git://github.com/your-user/your-awesome-wordpress-project.git ~/Sites/
↪your-awesome-wordpress-project
$ cd ~/Sites/your-awesome-wordpress-project
$ reward env-init your-awesome-wordpress-project
```

2. Before running the bootstrap command, you should import the WordPress database to the DB Container. To do so, first start the DB container:

```
$ reward env up -- db
```

3. Import the database.

```
$ reward db import < /path/to/db-dump-wordpress.sql
```

4. When the import is done, you can run the bootstrap.

```
$ reward bootstrap
```

1.3.10 Initializing a Custom Node Environment in a Subdomain

First, create our node "application":

`vim app.js`

```
const http = require('http');

const hostname = '0.0.0.0';
const port = 3000;

const server = http.createServer((req, res) => {
    res.statusCode = 200;
    res.setHeader('Content-Type', 'text/plain');
    res.end('Hello World');
});

server.listen(port, hostname, () => {
    console.log(`Server running at http://${hostname}:${port}/`);
});
```

Next create Reward's `.env` file in the same directory and modify it as you wish.

```
reward env-init webapp --environment-type=local
```

With the first 3 variables you can configure the target container, the shell and the user of `reward shell`.

With the next 3 variables you can configure the target of the synchronization (to which container, to which folder) and you can disable sync if you want.

`vim .env`

```
REWARD_SHELL_CONTAINER=custom-container
REWARD_SHELL_COMMAND=sh
REWARD_SHELL_USER=node

REWARD_SYNC_ENABLED=true
REWARD_SYNC_CONTAINER=custom-container
REWARD_SYNC_PATH=/app

REWARD_ENV_NAME=webapp
```

(continues on next page)

(continued from previous page)

```
REWARD_ENV_TYPE=local
REWARD_WEB_ROOT=/

TRAEFIK_DOMAIN=webapp.test
TRAEFIK_SUBDOMAIN=custom
TRAEFIK_EXTRA_HOSTS=

REWARD_DB=false
REWARD_ELASTICSEARCH=false
REWARD_OPENSEARCH=false
REWARD_VARNISH=false
REWARD_RABBITMQ=false
REWARD_REDIS=false
REWARD_MERCURE=false
```

Now we define the new environment using a custom go template.

Note: if you modify the container name in the service you should modify it in the labels as well!

`vim .reward/reward-env.yml`

```
version: "3.5"
services:
  custom-container:
    hostname: "{{ $.reward_env_name }}-node"
    build:
      context: .
      dockerfile: .reward/Dockerfile
    volumes:
      - appdata:/app
    extra_hosts:
      - {{ $.traefik_domain }}:{{ default "0.0.0.0" $.traefik_address }}
      - {{ default "app" $.traefik_subdomain }}.{{ $.traefik_domain }}:{{ default "0.0.0.
↪0" $.traefik_address}}
    labels:
      - traefik.enable=true
      - traefik.http.routers.custom.tls=true
      - traefik.http.routers.custom.rule=Host(`{{ default "app" .traefik_subdomain }}.{{␣
↪default "custom.test" .traefik_domain }}`)
      - traefik.http.services.custom.loadbalancer.server.port=3000
      - traefik.docker.network={{ $.reward_env_name }}
      - dev.reward.container.name=custom-container
      - dev.reward.environment.name={{ $.reward_env_name }}

volumes:
  appdata: {}
```

And finally create a custom Dockerfile

`vim .reward/Dockerfile`

```
FROM node:lts
WORKDIR /app
COPY . /app
```

(continues on next page)

(continued from previous page)

```
ARG DOCKER_START_COMMAND="node app.js"
ENV DOCKER_START_COMMAND=$DOCKER_START_COMMAND

CMD ["sh", "-c", "while true; do ${DOCKER_START_COMMAND}; sleep 10; done"]
```

1.4 Usage

1.4.1 Useful Commands

- Print information about the environments.

```
reward info
```

- Run only the db container

```
reward env up -- db
```

- Launch a shell session within the project environment's php-fpm container:

```
reward shell
```

```
# or launch an `sh` shell in the nginx container
reward shell sh --container nginx
```

- Start a stopped environment:

```
reward env start
```

- Stop a running environment:

```
reward env stop
```

- Forcefully recreate a container:

```
reward env up --force-recreate --no-deps php-fpm
```

- Remove the environment and volumes completely:

```
reward env down -v
```

- Import a database:

```
# for plain SQL database dump you can simply use os' stdin
reward db import < /path/to/dump.sql
```

```
# for compressed database dump
gunzip /path/to/dump.sql.gz -c | reward db import
```

Note: If you face some weird issues during the database import, you can try to increase the line buffer. By default it's 10 MB.

reward db import –line-buffer 50 < /path/to/dump.sql

- Pass additional flags to MySQL during the import.

  Note the "empty" *double dashes* (`--`) here. All the stuff after them will be passed to MySQL.

```
# mysql --force
reward db import -- --force
```

- Run complex MySQL queries directly using `reward db connect`:

```
# Note: to pass arguments use double dash to terminate Reward's argument parsing and␣
↪escape the special characters [;'"]*
# Run inline query:
$ reward db connect -- -e \"SELECT table_name FROM information_schema.tables WHERE␣
↪table_schema=\'magento\' ORDER BY table_name LIMIT 5\;\"

# Run query passing a bash variable (note the escaped quote):
$ MYSQL_CMD="\"SELECT table_name FROM information_schema.tables WHERE table_schema=
↪'magento' ORDER BY table_name LIMIT 5;\""

$ reward db connect -- -e $MYSQL_CMD

# Run multiple queries/commands using heredoc:
$ MYSQL_CMD=$(cat <<"EOF"
"SELECT table_name FROM information_schema.tables WHERE table_schema='magento'␣
↪ORDER BY table_name LIMIT 5;
QUERY2;
QUERY3;"
EOF
)

$ reward db connect -- -e $MYSQL_CMD
```

- Dump database:

```
reward db dump | gzip -c > /path/to/db-dump.sql.gz
```

- Connect database using root user:

```
reward db connect --root
```

- Monitor database processlist:

```
watch -n 3 "reward db connect -- -e \'show processlist\'"
```

- Tail environment nginx and php logs:

```
reward env logs --tail 0 -f nginx php-fpm php-debug
```

- Tail the varnish activity log:

```
reward env exec -T varnish varnishlog
```

- Clean varnish cache:

```
reward env exec varnish varnishadm 'ban req.url ~ .'
```

```
# or you can use this, these commands are identical:
reward shell --container varnish varnishadm 'ban req.url ~ .'
```

- Connect to redis:

```
reward env exec redis redis-cli
```

- Flush redis completely:

```
reward env exec -T redis redis-cli flushall
```

Further Information

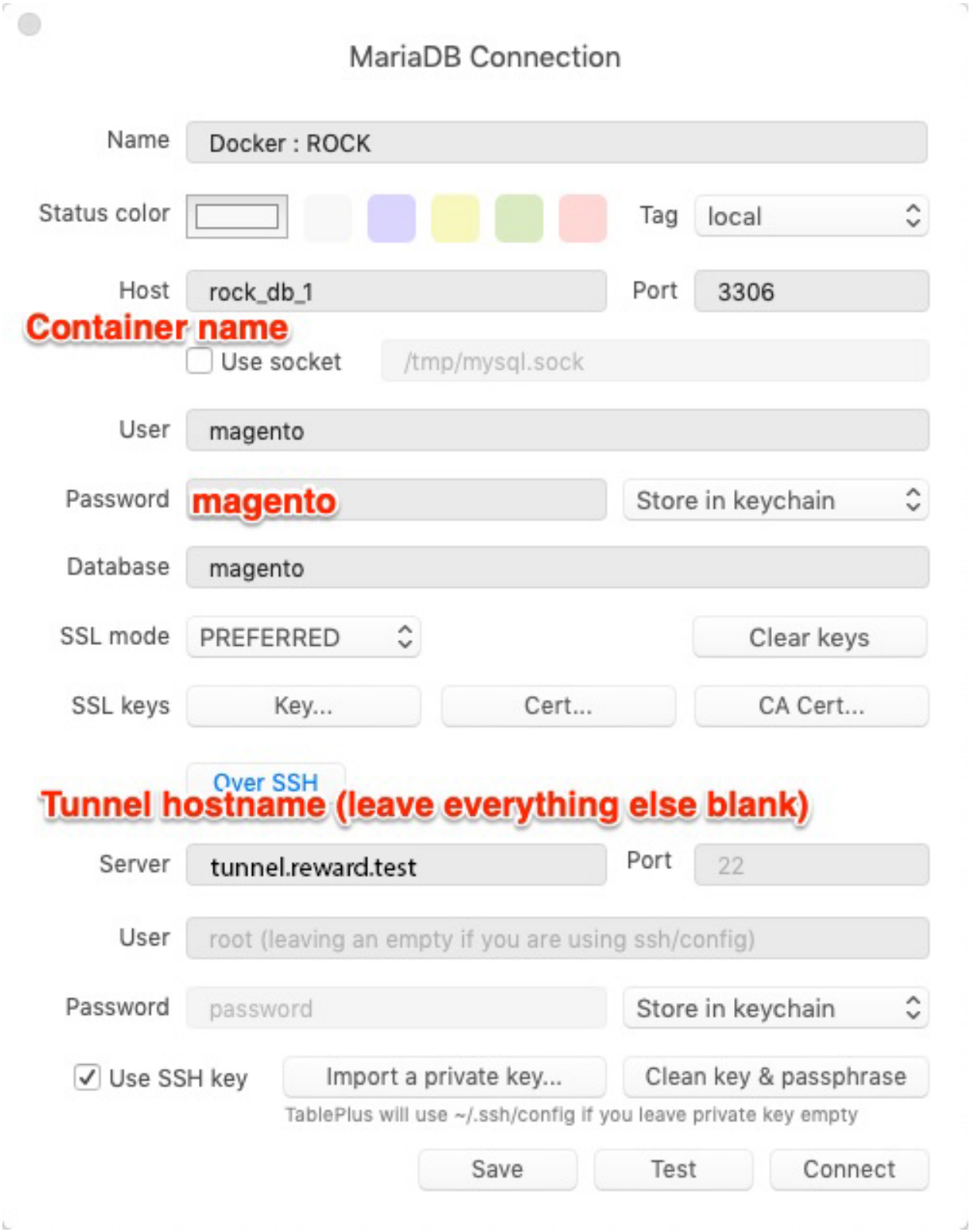
You can call `--help` for any of reward's commands. For example `reward --help` or `reward env --help` for more details and useful command information.

1.4.2 Connect to the Database

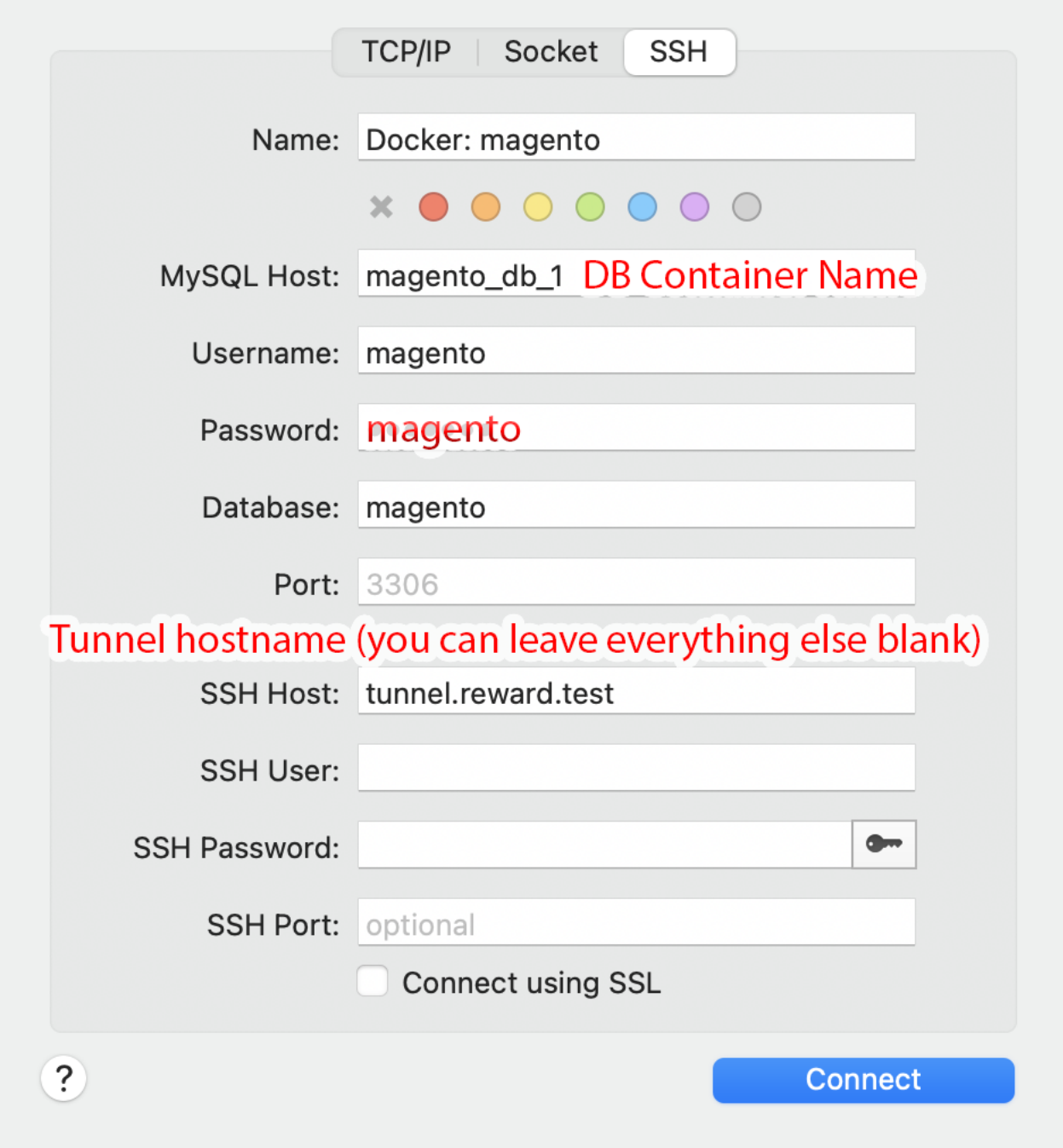
Common Settings

Note: On Windows use "Key pair Authentication" and select the SSH private key from Reward's home directory. See the example above.

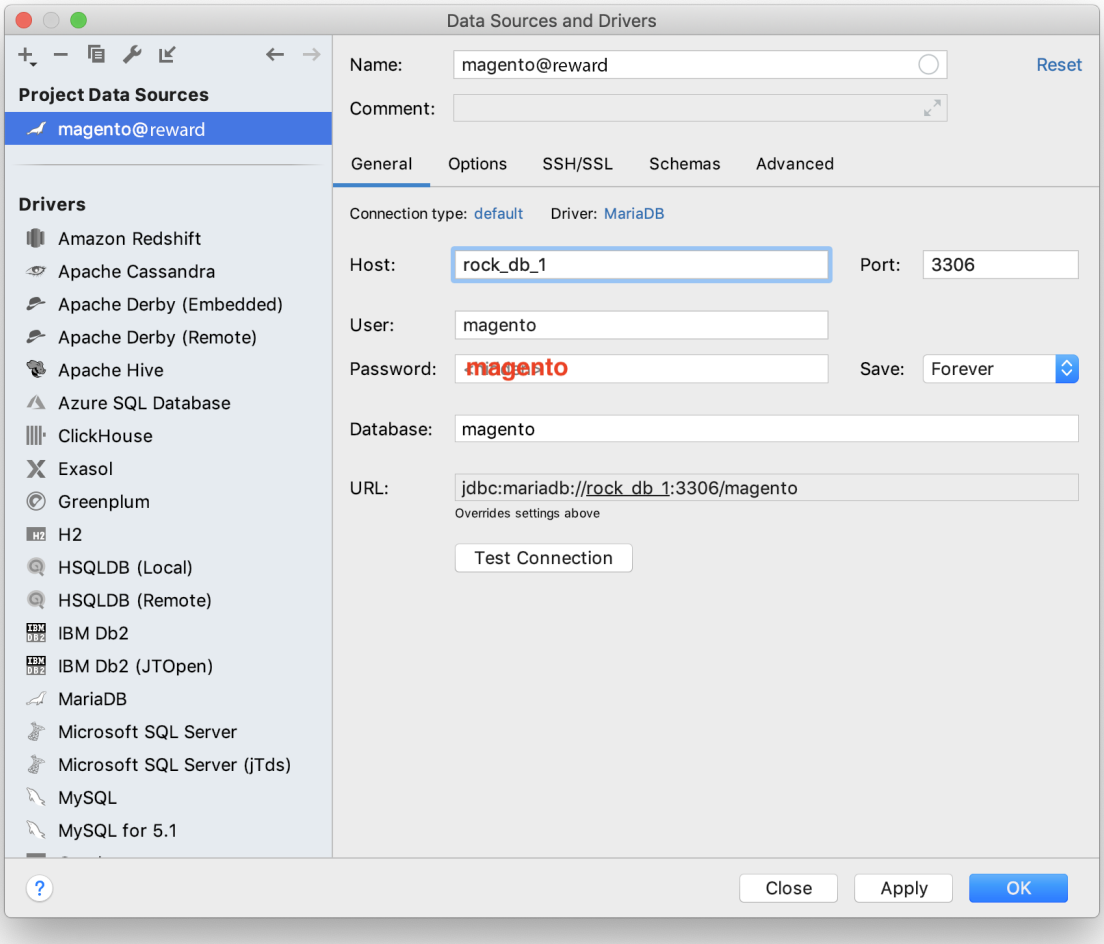
TablePlus

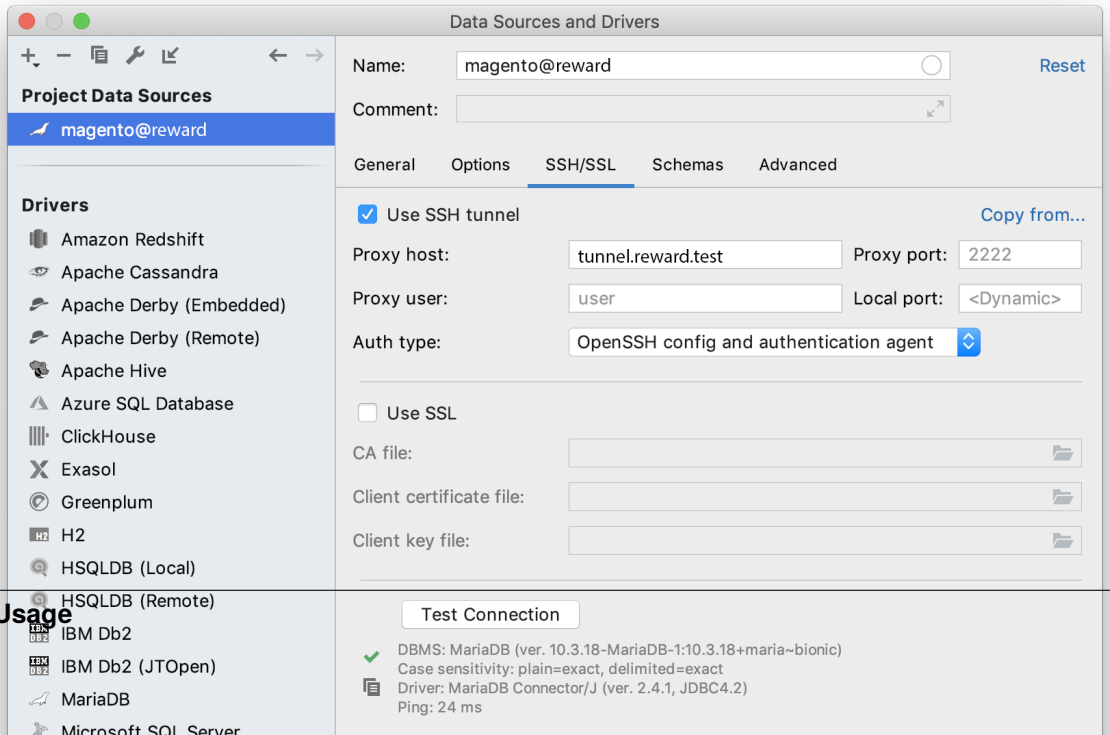

MariaDB Connection
Name
Docker : ROCK
Status color
Tag
local
Host
rock_db_1
Port
3306
Container name
Use socket
/tmp/mysql.sock
User
magento
Password
magento
Store in keychain
Database
magento
SSL mode
PREFERRED
Clear keys
SSL keys
Key...
Cert...
CA Cert...
Over SSH
Tunnel hostname (leave everything else blank)
Server
tunnel.reward.test
Port
22
User
root (leaving an empty if you are using ssh/config)
Password
password
Store in keychain
Use SSH key
Import a private key...
Clean key & passphrase
TablePlus will use ~/.ssh/config if you leave private key empty
Save
Test
Connect

Sequel Pro / Sequel Ace

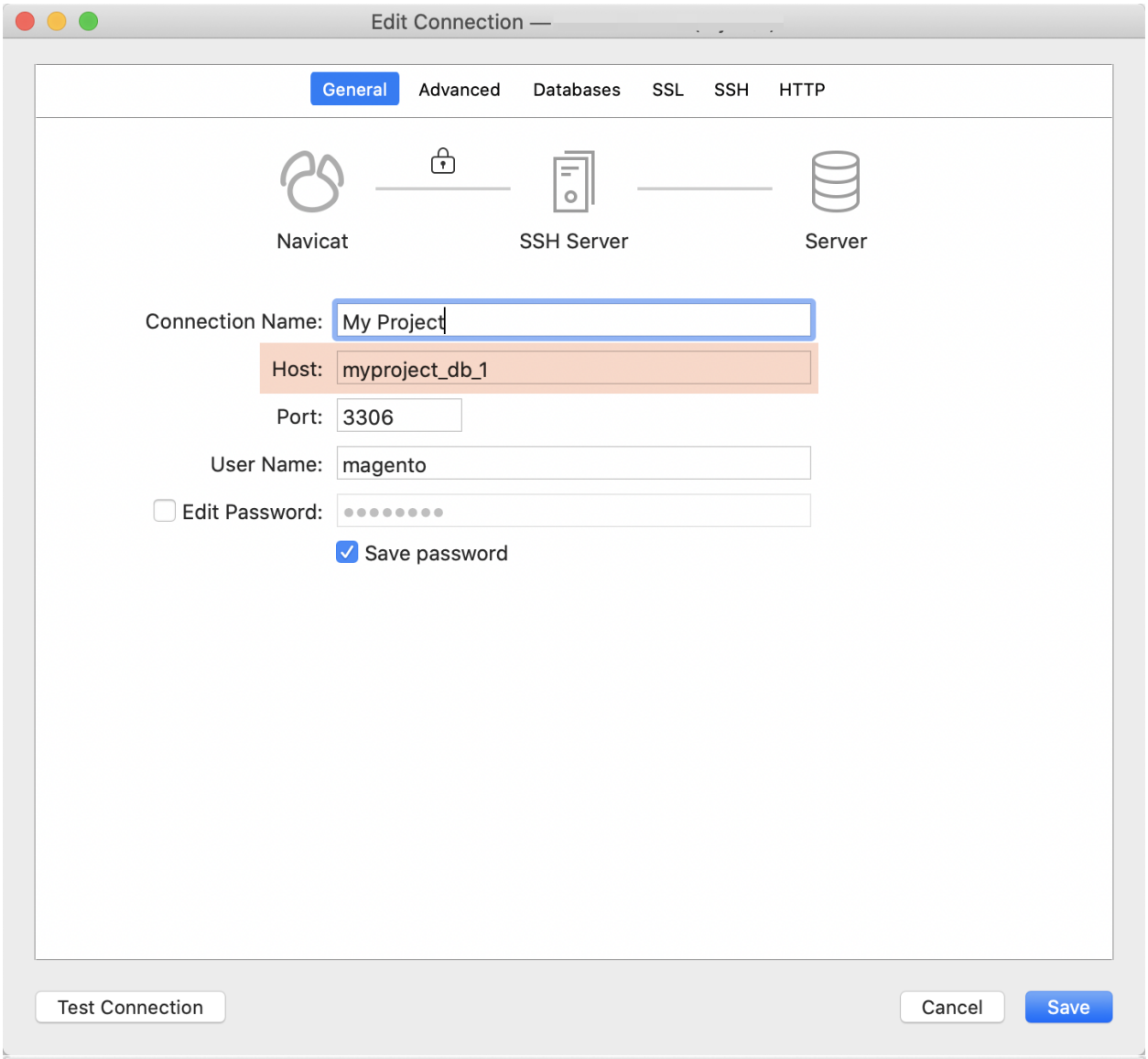


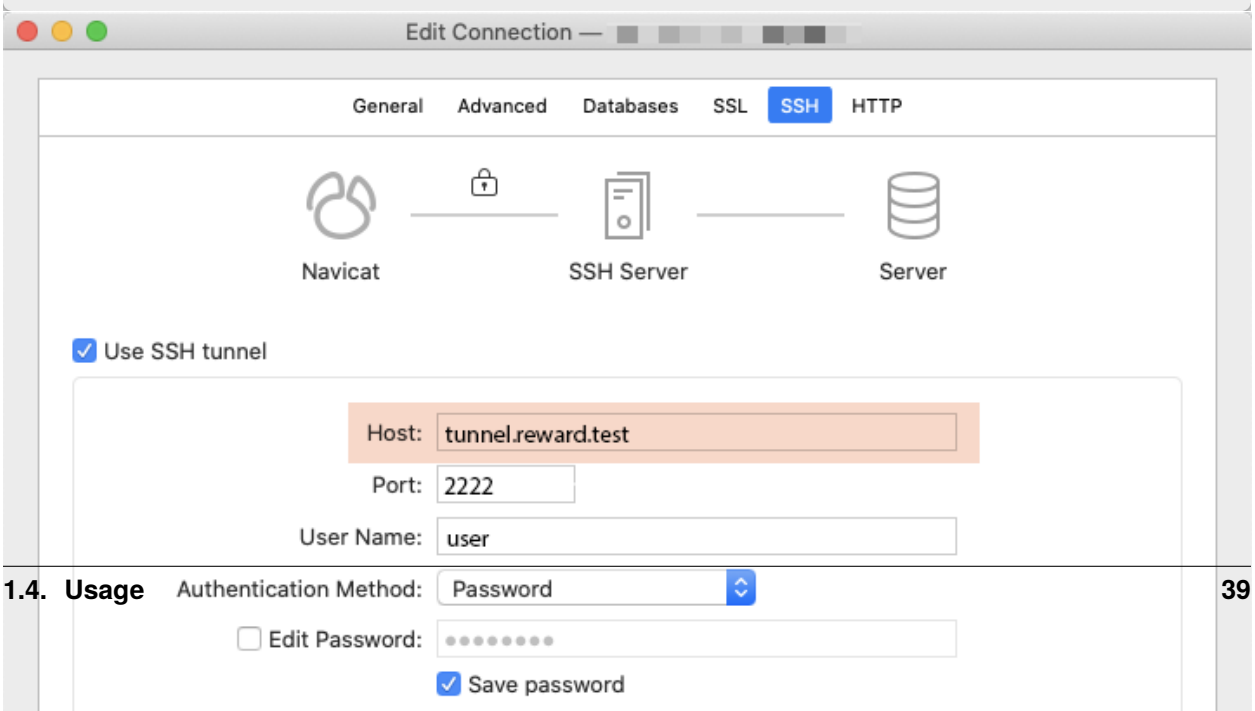
PhpStorm


Data Sources and Drivers
Project Data Sources
magento@reward
Drivers
Amazon Redshift
Apache Cassandra
Apache Derby (Embedded)
Apache Derby (Remote)
Apache Hive
Azure SQL Database
ClickHouse
Exasol
Greenplum
H2
HSQLDB (Local)
HSQLDB (Remote)
IBM Db2
IBM Db2 (JTOpen)
MariaDB
Microsoft SQL Server
Microsoft SQL Server (jTds)
MySQL
MySQL for 5.1
Name: magento@reward
Reset
Comment:
General
Options
SSH/SSL
Schemas
Advanced
Connection type: default
Driver: MariaDB
Host: rock_db_1
Port: 3306
User: magento
Password: magento
Save: Forever
Database: magento
URL: jdbc:mariadb://rock_db_1:3306/magento
Overrides settings above
Test Connection
Close
Apply
OK


Data Sources and Drivers
Project Data Sources
magento@reward
Drivers
Amazon Redshift
Apache Cassandra
Apache Derby (Embedded)
Apache Derby (Remote)
Apache Hive
Azure SQL Database
ClickHouse
Exasol
Greenplum
H2
HSQLDB (Local)
HSQLDB (Remote)
IBM Db2
IBM Db2 (JTOpen)
MariaDB
Name: magento@reward
Reset
Comment:
General
Options
SSH/SSL
Schemas
Advanced
Use SSH tunnel
Copy from...
Proxy host: tunnel.reward.test
Proxy port: 2222
Proxy user: user
Local port: <Dynamic>
Auth type: OpenSSH config and authentication agent
Use SSL
CA file:
Client certificate file:
Client key file:
Test Connection
DBMS: MariaDB (ver. 10.3.18-MariaDB-1:10.3.18+maria~bionic)
Case sensitivity: plain=exact, delimited=exact
Driver: MariaDB Connector/J (ver. 2.4.1, JDBC4.2)
Ping: 24 ms

Navicat for MySQL





1.5 Customization

1.5.1 Additional Packages

You can override the service containers and install additional packages to them using the following method.

Override service

First, create an **override file** for the service you want to modify. For example, if you want to modify the `php-fpm` service, create a file named `reward-env.yml` in the `.reward` directory.

Then, add the following snippet to the override file:

`vim .reward/reward-env.yml`

```
version: "3.5"
services:
  php-fpm:
    build:
      context: .
      dockerfile: .reward/Dockerfile
```

Dockerfile

Next, create the **custom Dockerfile**. The one below uses the default php-fpm 7.4 container for magento2 environments.

Note: If you want to see what image is used by your current environment, run the following command:

reward env config

`vim .reward/Dockerfile`

```
FROM docker.io/rewardenv/php-fpm:7.4-magento2

USER root

RUN apt-get update && apt-get install -y --no-install-recommends \
    telnet \
    && rm -rf /var/lib/apt/lists/*

USER www-data
```

Now we are ready, let's rebuild the environment.

```
reward env up --build
```

1.5.2 Composer

Composer home

Composer home `~/.composer` directory is mounted and shared from your host system to the `php-fpm` container. This makes possible to share Composer's cache, and your Composer auth configuration between environments.

Change Composer version by environment

From Reward >0.2.0 it is possible to configure `COMPOSER_VERSION` in the .env file like this:

```
COMPOSER_VERSION=2
```

Default Composer versioning matrix by environment type:

Change Composer version interactively inside the Reward Shell

```
$ reward shell
$ sudo alternatives --config composer
```

1.5.3 Custom SSL Certificates

If you need additional domains with custom certificates for your project, you can do it by following this guide.

Let's assume you want to use the `custom-domain.com` domain for your project.

1. Add the custom domain to your `.env` file. You can add multiple domains separated by space.

   ```
   TRAEFIK_EXTRA_HOSTS="custom-domain.com"
   ```

2. Copy the certificate and the certificate key to the `~/.reward/ssl/certs` directory:

   ```
   cp ~/Downloads/key.pem ~/.reward/ssl/certs/custom-domain.com.key.pem
   cp ~/Downloads/cert.pem ~/.reward/ssl/certs/custom-domain.com.crt.pem
   ```

3. **OPTIONAL**: The hosts from the `TRAEFIK_EXTRA_HOSTS` will be automatically configured and mapped to the webservers.

   However, if you want to finetune the domains (eg: you want to add wildcard domains) create a `.reward/reward-env.yml` file with the contents below (this will be additive to the docker-compose config Reward uses for the env, anything added here will be merged in, and you can see the complete config using `reward env config`):

   ```
   version: "3.5"
   services:
     varnish:
       labels:
         - traefik.http.routers.{{.reward_env_name}}-varnish.rule=
             HostRegexp(`{subdomain:.+}.{{.traefik_domain}}`)
             || Host(`{{.traefik_domain}}`)
             || HostRegexp(`{subdomain:.+}.custom-domain.com`)
             || Host(`custom-domain.com`)
   ```

(continues on next page)

(continued from previous page)

```
  nginx:
    labels:
      - traefik.http.routers.{{.reward_env_name}}-nginx.rule=
          HostRegexp(`{subdomain:.+}.{{.traefik_domain}}`)
          || Host(`{{.traefik_domain}}`)
          || HostRegexp(`{subdomain:.+}.custom-domain.com`)
          || Host(`custom-domain.com`)
```

4. Bring up the environment

```
reward env up
```

5. Restart Traefik

```
reward svc restart traefik
```

1.5.4 Customizing An Environment

Further information on customizing or extending an environment is forthcoming. For now, this section is limited to very simple and somewhat common customizations.

To configure your project with a non-default PHP version, add the following to the project's `.env` file and run `reward env up` to re-create the affected containers:

```
PHP_VERSION=7.4
```

The versions of MariaDB, Elasticsearch, Varnish, Redis, and NodeJS may also be similarly configured using variables in the `.env` file:

- `MARIADB_VERSION`
- `ELASTICSEARCH_VERSION`
- `REDIS_VERSION`
- `VARNISH_VERSION`
- `RABBITMQ_VERSION`
- `NODE_VERSION`

The components in an environment can be skipped by disabling these variables in `.env` file:

- `REWARD_DB=false`
- `REWARD_REDIS=false`

Customize a Reward environment to be able to reach another Reward environment

To make it possible to reach another Reward environment, the container DNS have to resolve the other project's domain (eg.: `otherproject.test`) to Reward's Traefik container.

To do so add a space separated list of domains to the TRAEFIK_EXTRA_HOSTS variable in the .env file.

- `TRAEFIK_EXTRA_HOSTS="otherproject.test thirdproject.test"`

1.5.5 Database

You can change the DB Server Character Set or Collation in the `.env` file:

- `MYSQL_CHARACTER_SET_SERVER=utf8mb4`
- `MYSQL_COLLATION_SERVER=utf8mb4_unicode_ci`

To configure InnoDB Buffer Pool size, add the following line to the `.env` file:

- `MYSQL_INNODB_BUFFER_POOL_SIZE=256m`

To disable Strict Mode in MySQL you will have to add the following line to the `.env` file:

- `MYSQL_DISABLE_STRICT_MODE=1`

You can also set additional arguments to `mysqld` using the setting below in the `.env` file. To add multiple arguments, set them as a single **space separated** string. See the available arguments in the MariaDB Docs.

- `MYSQL_ARGS="--innodb-buffer-pool-instances=4 --key-buffer-size=256M"`

1.5.6 Magento 2

The following variables can be added to the project's `.env` file to enable additional database containers for use with the Magento 2 (Commerce Only) split-database solution.

- `REWARD_SPLIT_SALES=true`
- `REWARD_SPLIT_CHECKOUT=true`

Some unnecessary Magento 2 specific components can be disabled by using these environment variables in `.env` file:

- `REWARD_ELASTICSEARCH=false`
- `REWARD_OPENSEARCH=false`
- `REWARD_VARNISH=false`
- `REWARD_RABBITMQ=false`

1.5.7 Mercure

For information on what Mercure is, please see the introduction to Mercure in Mercure documentation.

Mercure can be enabled in all environment types by changing the following to the project's `.env` file (or exporting them to environment variables prior to starting the environment):

`REWARD_MERCURE=true` -> `REWARD_MERCURE=true`

The following variables have predefined values and those can be changed optionally in the `.env` file:

- `SERVER_NAME=":80"`

- MERCURE_PUBLISHER_JWT_KEY="password"
- MERCURE_PUBLISHER_JWT_ALG="HS256"
- MERCURE_SUBSCRIBER_JWT_KEY="password"
- MERCURE_SUBSCRIBER_JWT_ALG="HS256"
- MERCURE_EXTRA_DIRECTIVES=""

1.5.8 Multiple Domains

If you need multiple domains configured for your project, Reward will now automatically route all sub-domains of the configured TRAEFIK_DOMAIN (as given when running env-init) to the Varnish/Nginx containers provided there is not a more specific rule such as for example rabbitmq.exampleproject.com which routes to the rabbitmq service for the project.

Multiple top-level domains may also be setup by following the instructions below:

1. Sign certificates for your new domains:

```
reward sign-certificate alternate1.test
reward sign-certificate alternate2.test
```

2. **OPTIONAL**: The hosts from the TRAEFIK_EXTRA_HOSTS will be automatically configured and mapped to the webservers.

   It's possible to add additional host routing rules. Create a .reward/reward-env.yml file with the contents below (this will be additive to the docker-compose config Reward uses for the env, anything added here will be merged in, and you can see the complete config using reward env config):

```
version: "3.5"
services:
  varnish:
    labels:
      - traefik.http.routers.{{.reward_env_name}}-varnish.rule=
          HostRegexp(`{subdomain:.+}.{{.traefik_domain}}`)
          || Host(`{{.traefik_domain}}`)
          || HostRegexp(`{subdomain:.+}.alternate1.test`)
          || Host(`alternate1.test`)
          || HostRegexp(`{subdomain:.+}.alternate2.test`)
          || Host(`alternate2.test`)
  nginx:
    labels:
      - traefik.http.routers.{{.reward_env_name}}-nginx.rule=
          HostRegexp(`{subdomain:.+}.{{.traefik_domain}}`)
          || Host(`{{.traefik_domain}}`)
          || HostRegexp(`{subdomain:.+}.alternate1.test`)
          || Host(`alternate1.test`)
          || HostRegexp(`{subdomain:.+}.alternate2.test`)
          || Host(`alternate2.test`)
```

3. Configure the application to handle traffic coming from each of these domains appropriately. An example on this for Magento 2 environments may be found below.
4. Run reward env up -d to update the containers, after which each of the URLs should work as expected.

Note: If these alternate domains must be resolvable from within the FPM containers, you must also leverage `extra_hosts` to add each specific sub-domain to the `/etc/hosts` file of the container as dnsmasq is used only on the host machine, not inside the containers. This should look something like the following excerpt.

From Reward >0.2.0 it is possible to add additional domains using `TRAEFIK_EXTRA_HOSTS` variable.

```
TRAEFIK_EXTRA_HOSTS="alternate1.test sub1.alternate1.test alternate2.test and.test␣
↪so.test on.test"
```

Before Reward 0.2.0 you have to add these lines to the `.reward/reward-env.yml` file as you did in step 2.

```
version: "3.5"
services:
  php-fpm:
   extra_hosts:
     - alternate1.test:{{default "0.0.0.0" .traefik_address}}
     - sub1.alternate1.test:{{default "0.0.0.0" .traefik_address}}
     - sub2.alternate1.test:{{default "0.0.0.0" .traefik_address}}
     - alternate2.test:{{default "0.0.0.0" .traefik_address}}
     - sub1.alternate2.test:{{default "0.0.0.0" .traefik_address}}
     - sub2.alternate2.test:{{default "0.0.0.0" .traefik_address}}

  php-debug:
   extra_hosts:
     - alternate1.test:{{default "0.0.0.0" .traefik_address}}
     - sub1.alternate1.test:{{default "0.0.0.0" .traefik_address}}
     - sub2.alternate1.test:{{default "0.0.0.0" .traefik_address}}
     - alternate2.test:{{default "0.0.0.0" .traefik_address}}
     - sub1.alternate2.test:{{default "0.0.0.0" .traefik_address}}
     - sub2.alternate2.test:{{default "0.0.0.0" .traefik_address}}
```

Magento Run Params (eg. Magento Multi Store)

There are two (and many more) ways to configure Magento run params (`MAGE_RUN_TYPE`, `MAGE_RUN_CODE`).

- Nginx mappings
- Composer autoload

Nginx host mappings

Nginx makes it possible to map values to variables based on other variable's values (eg: set a variable based on the hostname).

There are two ways to configure this:

1. **Using environment variables in the `.env` file**

   You can specify up to 99 mappings using the `NGINX_HOST_MAPPING_0` to `NGINX_HOST_MAPPING_99` environment variables. The variable will be split on `:` and the first value will be used as the hostname, the second value will be used as the `MAGE_RUN_CODE` and the third value will be used as the `MAGE_RUN_TYPE`.

```
NGINX_HOST_MAPPING_0="example.test:default:store"
NGINX_HOST_MAPPING_1="sub.example.test:store_code_1:store"
NGINX_HOST_MAPPING_2="website.example.test:another_run_code:website"
NGINX_HOST_MAPPING_99="default:default:store"
```

This will generate the same config as the example below.

If the *default* hostname is not specified it will be added automatically.

2. **Extending the nginx config files**

   Example: Add the following file to you project folder `./.reward/nginx/http-maps.conf` with the content below.

   Don't forget to restart your nginx container. `reward env restart -- nginx`

   - if the `$http_host` value is `sub.example.test`, nginx will map value `store_code_1` to `$MAGE_RUN_CODE`.
   - if the `$http_host` value is `sub.example.test`, nginx will map value `store` to `$MAGE_RUN_TYPE`.

```
map $http_host $MAGE_RUN_CODE {
    example.test            default;
    sub.example.test        store_code_1;
    website.example.test    another_run_code;
    default                 default;
}
map $http_host $MAGE_RUN_TYPE {
    example.test            store;
    sub.example.test        store;
    website.example.test    website;
    default                 store;
}
```

Composer autoload php file

When multiple domains are being used to load different stores or websites on Magento 2, the following configuration should be defined in order to set run codes and types as needed.

1. Add a file at `app/etc/stores.php` with the following contents:

```
<?php

use \Magento\Store\Model\StoreManager;
$serverName = isset($_SERVER['HTTP_HOST']) ? $_SERVER['HTTP_HOST'] : null;

switch ($serverName) {
    case 'domain1.exampleproject.test':
        $runCode = 'examplecode1';
        $runType = 'website';
        break;
    case 'domain2.exampleproject.test':
        $runCode = 'examplecode2';
        $runType = 'website';
        break;
```

(continues on next page)

(continued from previous page)

```
    default:
        return;
}

if ((!isset($_SERVER[StoreManager::PARAM_RUN_TYPE])
        || !$_SERVER[StoreManager::PARAM_RUN_TYPE])
    && (!isset($_SERVER[StoreManager::PARAM_RUN_CODE])
        || !$_SERVER[StoreManager::PARAM_RUN_CODE])
) {
    $_SERVER[StoreManager::PARAM_RUN_CODE] = $runCode;
    $_SERVER[StoreManager::PARAM_RUN_TYPE] = $runType;
}
```

Note: The above example will not alter production site behavior given the default is to return should the `HTTP_HOST` value not match one of the defined `case` statements. This is desired as some hosting environments define run codes and types in an Nginx mapping. One may add production host names to the switch block should it be desired to use the same site switching mechanism across all environments.

---

2. Then in `composer.json` add the file created in the previous step to the list of files which are automatically loaded by composer on each web request:

```
{
    "autoload": {
        "files": [
            "app/etc/stores.php"
        ]
    }
}
```

Note: This is similar to using *magento-vars.php* on Magento Commerce Cloud, but using composer to load the file rather than relying on Commerce Cloud magic: https://devdocs.magento.com/guides/v2.3/cloud/project/project-multi-sites.html

---

3. After editing the `composer.json` regenerate the autoload configuration:

```
composer dump-autoload
```

1.5.9 Nginx Configuration

It is possible to use custom nginx configurations in various ways. When you run `reward env up` it will map `./.reward/nginx` directory to the container under `/etc/nginx/snippets` directory.

Nginx Application template

You can override Nginx HTTP and Server blocks as well.

Nginx default.conf looks like this by default:

```
include /etc/nginx/snippets/http-*.conf;

server {
    listen 80;

    root ${NGINX_ROOT}${NGINX_PUBLIC};
    set $MAGE_ROOT ${NGINX_ROOT};

    index index.html index.php;
    autoindex off;
    charset UTF-8;

    include /etc/nginx/snippets/server-*.conf;
    include /etc/nginx/available.d/${NGINX_TEMPLATE};
}
```

Using this it is possible to inject configurations in 2 places.

Note: If you'd like to create mappings (which have to be under nginx HTTP block) you can define templates in the *./.reward/nginx* directory using *http-*.conf* pattern. For example:

```
./.reward/nginx/http-example-mappings.conf
```

Note: If you'd like to create redirects (which have to be under nginx servers block) you can define templates in the *./.reward/nginx* directory using *server-*.conf* pattern. For example:

```
./.reward/nginx/server-example-redirects.conf
```

Example: Create a rewrite from example.test/blog/* to blog.example.test/*

Create a file under the `./.reward/nginx` directory called server-redirect.conf.

```
$ echo -e 'location ^~ /blog/ {
    rewrite ^/blog/(.*) https://blog.$http_host/$1 permanent;
  }' > ./.reward/nginx/server-redirect.conf

$ reward env restart -- nginx
```

(continues on next page)

(continued from previous page)

```
# Test using curl
$ curl -IL https://example.test/blog/test-url
HTTP/2 301
content-type: text/html
date: Fri, 29 Jan 2021 14:12:22 GMT
location: https://blog.example.test/test-url
server: nginx/1.16.1
content-length: 169
```

1.5.10 PHP

The following PHP environment variables are set to these values and optionally can be updated in the `.env` file:

- `PHP_EXPOSE=Off`
- `PHP_ERROR_REPORTING=E_ALL`
- `PHP_DISPLAY_ERRORS=On`
- `PHP_DISPLAY_STARTUP_ERRORS=On`
- `PHP_LOG_ERRORS=On`
- `PHP_LOG_ERRORS_MAX_LEN=1024`
- `PHP_MAX_EXECUTION_TIME=3600`
- `PHP_MAX_INPUT_VARS=10000`
- `PHP_POST_MAX_SIZE=64M`
- `PHP_UPLOAD_MAX_FILESIZE=64M`
- `PHP_MAX_FILE_UPLOADS=20`
- `PHP_MEMORY_LIMIT=4G`
- `PHP_SESSION_AUTO_START=Off`
- `PHP_REALPATH_CACHE_SIZE=10M`
- `PHP_REALPATH_CACHE_TTL=7200`
- `PHP_DATE_TIMEZONE=UTC`
- `PHP_ZEND_ASSERTIONS=1`
- `PHP_SENDMAIL_PATH="/usr/sbin/sendmail -t -i"`
- `PHP_OPCACHE_ENABLE=On`
- `PHP_OPCACHE_MEMORY_CONSUMPTION=512`

1.5.11 Search Engine

OpenSearch or Elasticsearch

Reward currently supports both Elasticsearch and OpenSearch as Search Engines.

To use one of them, you'll need to enable one of them the `.env` file:

- `REWARD_ELASTICSEARCH=false`
- `REWARD_OPENSEARCH=true`

If you enable both, Reward will install Magento using OpenSearch.

OpenSearch Dashboards

Reward also supports OpenSearch Dashboards. Enable it in the `.env` file:

- `REWARD_OPENSEARCH_DASHBOARDS=true`

It is not a global service. You can reach it as a subdomain of the development url:

`https://opensearch-dashboards.projectname.test`

OpenSearch Configuration

You can change the OpenSearch version by changing it in the `.env` file. The available version can be found here.

- `OPENSEARCH_VERSION=1.2`

You can also configure the memory limitations for OpenSearch.

- `OPENSEARCH_XMS=64m`
- `OPENSEARCH_XMX=512m`

Elasticsearch Configuration

You can change the Elasticsearch version by changing it in the `.env` file. The available version can be found here.

- `ELASTICSEARCH_VERSION=7.16`

You can also configure the memory limitations for Elasticsearch.

- `ELASTICSEARCH_XMS=64m`
- `ELASTICSEARCH_XMX=512m`

1.6 Advanced Configuration

1.6.1 Reward Settings

During the installation of Reward a global configuration file will be created in the user's HOME directory, called `~/.reward.yml`. It is possible to configure various settings of Reward in this configuration file, like what container image should Reward use for global services, what image repo should it use, etc.

Available settings

Set the logging level. Default is `info`.

- `log_level: info` - valid options: `trace`, `debug`, `info`, `warning`, `error`.

Enable debug logging (set log level to debug).

- `debug: false` - valid options: `false`, `true`

  Many options including this can be configured using an environment variable.

```
DEBUG=true reward env up
```

Disable default common services. These services are enabled by default.

- `reward_portainer: true` - valid options: `false`, `true`
- `reward_dnsmasq: true` - valid options: `false`, `true`
- `reward_tunnel: true` - valid options: `false`, `true`
- `reward_mailbox: true` - valid options: `false`, `true`
- `reward_phpmyadmin: true` - valid options: `false`, `true`
- `reward_elastichq: true` - valid options: `false`, `true`

Enable additional common services. These services are disabled by default.

- `reward_adminer: false` - valid options: `false`, `true`

Service Container Settings

It's possible to change service container images using the following vars.

- `reward_traefik_image: "traefik"`
- `reward_portainer_image: "portainer/portainer-ce"`
- `reward_dnsmasq_image: "docker.io/rewardenv/dnsmasq"`
- `reward_mailbox_image: "docker.io/rewardenv/mailbox:1.15"`

- `reward_tunnel_image: "docker.io/rewardenv/sshd"`
- `reward_phpmyadmin_image: "phpmyadmin"`
- `reward_elastichq_image: "elastichq/elasticsearch-hq"`
- `reward_adminer_image: "dehy/adminer"`

By default, Traefik listens on `0.0.0.0`. To change this behaviour you can add the following line to the config file and change the IP address to `127.0.0.1` to listen on localhost only. It is also possible to change the listening ports.

- `reward_traefik_listen: "127.0.0.1"`
- `reward_traefik_http_port: "80"`
- `reward_traefik_https_port: "443"`

You can also add additional http and https ports on top of the defaults (80, 443). This is useful when you want to expose a service on a different port than the default ones. See more info in the *Open Ports* section.

- `reward_traefik_bind_additional_http_ports: []` - valid option example: `[8080, 8081]`
- `reward_traefik_bind_additional_https_ports: []` - valid option example: `[8443, 9443]`

By default, Reward makes it possible to resolve the environment's domain to the nginx container's IP address inside the docker network. To disable this behaviour you add this line to the config file.

- `reward_resolve_domain_to_traefik: false`

By default, Reward redirects all http traffic to https. To disable this behaviour you add this line to the config file.

- `reward_traefik_allow_http: true`

Logging level of traefik (default: "info").

- `reward_traefik_log_level: info`

It is possible to change DNSMasq listen address and ports. By default, DNSMasq listens on `127.0.0.1` and on port `53`.

- `reward_dnsmasq_listen: "0.0.0.0"`
- `reward_dnsmasq_tcp_port: "53"`
- `reward_dnsmasq_udp_port: "53"`

By default, only the UDP port 53 is exposed from the dnsmasq container. Sometimes it doesn't seem to be enough, and the TCP port 53 has to be exposed as well. To do so enable the `reward_dnsmasq_bind_tcp` variable in the ~/.reward.yml file.

- `reward_dnsmasq_bind_tcp: false`
- `reward_dnsmasq_bind_udp: true`

It is possible to change Tunnel listen address and ports. By default, Tunnel listens on `0.0.0.0` and on port `2222`.

- `reward_tunnel_listen: "127.0.0.1"`
- `reward_tunnel_port: "2222"`

By default, Reward is not allowed to run commands as root. To disable this check you can add the following setting to the config.

- `reward_allow_superuser: true`

By default, Reward is going to use Mutagen sync for macOS and Windows. If you want to disable Mutagen you can set this in Reward config. Also, on Windows with WSL2 it's possible to use well performing direct mount from WSL2's drive. It is disabled by default. To enable this functionality, disable syncing with the following line to the config.

- `reward_sync_enabled: false`

Previously Reward used CentOS 7 based images, now the defaults are debian based images. Experimental images: `debian-bookworm`, `ubuntu-jammy`.

- `reward_docker_image_base: ubuntu-jammy`

By default Reward uses separated nginx + php-fpm containers. Enabling this setting will merge them to one "web" container.

- `reward_single_web_container: true`

1.6.2 Allure Reporting

Allure is a powerful **test reporting framework**. It is fully compatible with PHPUnit and Codeception. This means that each type of test available in **Magento 2** can be visualized.

Configuration

To enable Allure in your project environment, add to your `.env` file:

```
REWARD_ALLURE=true
```

As a result both `php-fpm` and `php-debug` containers get additional mount `/var/allure-results` where the Test results should be saved.

PHPUnit Reports

To visualize your PHPUnit Tests results *(Unit, Integration, API functional, Static)* in Allure, you need to edit `phpunit.xml` file and find `<listeners>` section. If you have `<listener class="Yandex\Allure\Adapter\AllureAdapter">` node, just change the path where logs are saved: `<string>/var/allure-results</string>` (please notice leading `/`).

```
    <listeners>
        <listener class="Magento\TestFramework\Event\PhpUnit"/>
        <listener class="Yandex\Allure\Adapter\AllureAdapter">
            <arguments>
                <string>/var/allure-results</string>
            </arguments>
        </listener>
    </listeners>
```

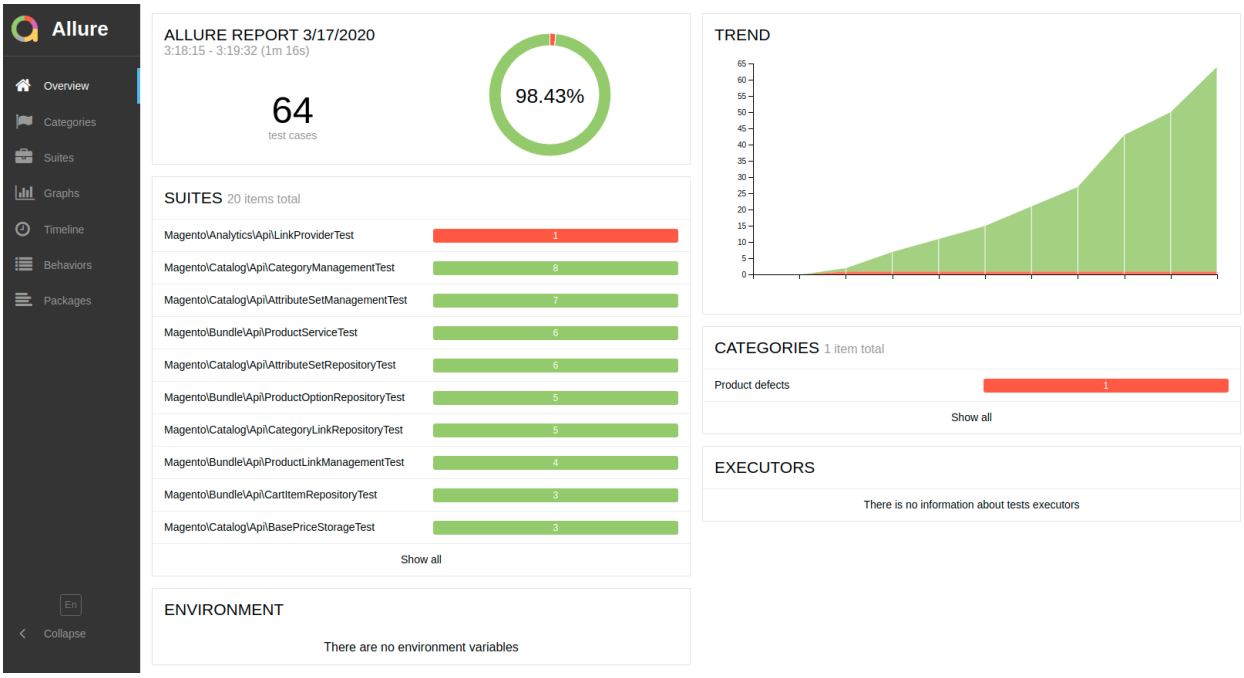
Codeception Reports

Adjusting Codeception reports path is a little more complicated. Please find the `dev/tests/acceptance/codeception.yml` file, under `Magento\FunctionalTestingFramework\Allure\Adapter\MagentoAllureAdapter` you'll find `outputDirectory`. Change its value to `/var/allure-results`.

```
extensions:
    config:
        Magento\FunctionalTestingFramework\Allure\Adapter\MagentoAllureAdapter:
            deletePreviousResults: false
            outputDirectory: /var/allure-results
```

Web Interface

Allure reports are available with your Web Browser on `allure` subdomain (e.g. `https://allure.magento2.test/`). Reports are generated with 5 seconds interval.



1.6.3 Automatic DNS Resolution

In order to allow automatic DNS resolution using the provided dnsmasq service we will need to make sure DNS request are routed through our local network. This requires some configuration.

Windows

NRPT Rule

On Windows you can set custom DNS for a specific domain using NRPT Rules. You must execute commands in your PowerShell console with admin privileges.

To add the necessary NRPT rule use the `Add-DnsClientNrptRule` command:

```
> Add-DnsClientNrptRule -Namespace ".test" -NameServers "127.0.0.1"
```

If you'd like to remove the rule, get the list of all the rules using `Get-DnsClientNrptRule`:

```
> Get-DnsClientNrptRule

Name                            : {RULE-NAME}
Version                         : 2
Namespace                       : {.test}
IPsecCARestriction              :
DirectAccessDnsServers          :
DirectAccessEnabled             : False
DirectAccessProxyType           :
DirectAccessProxyName           :
DirectAccessQueryIPsecEncryption :
DirectAccessQueryIPsecRequired  :
NameServers                     : 127.0.0.1
DnsSecEnabled                   : False
DnsSecQueryIPsecEncryption      :
DnsSecQueryIPsecRequired        :
DnsSecValidationRequired        :
NameEncoding                    : Disable
DisplayName                     :
Comment                         :
```

And then remove the rule using `Remove-DnsClientNrptRule`:

```
> Remove-DnsClientNrptRule -Name "{RULE-NAME}"
```

It also works if you're using WSL2.

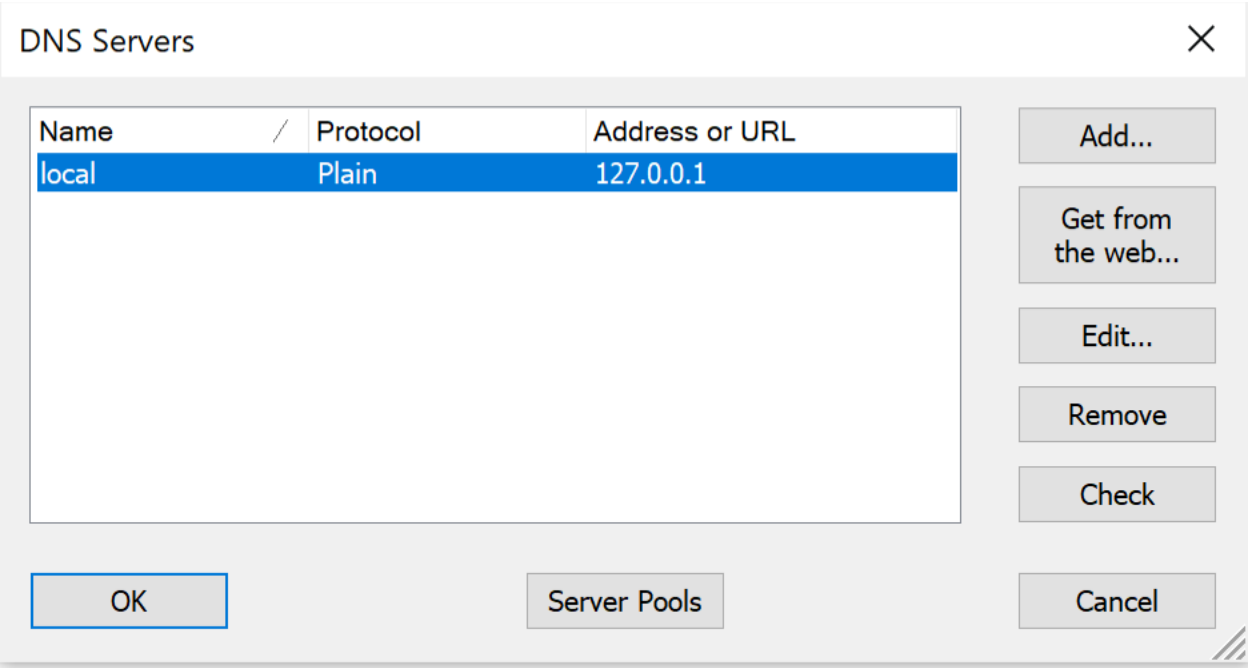
Source: https://superuser.com/a/1374119

YogaDNS

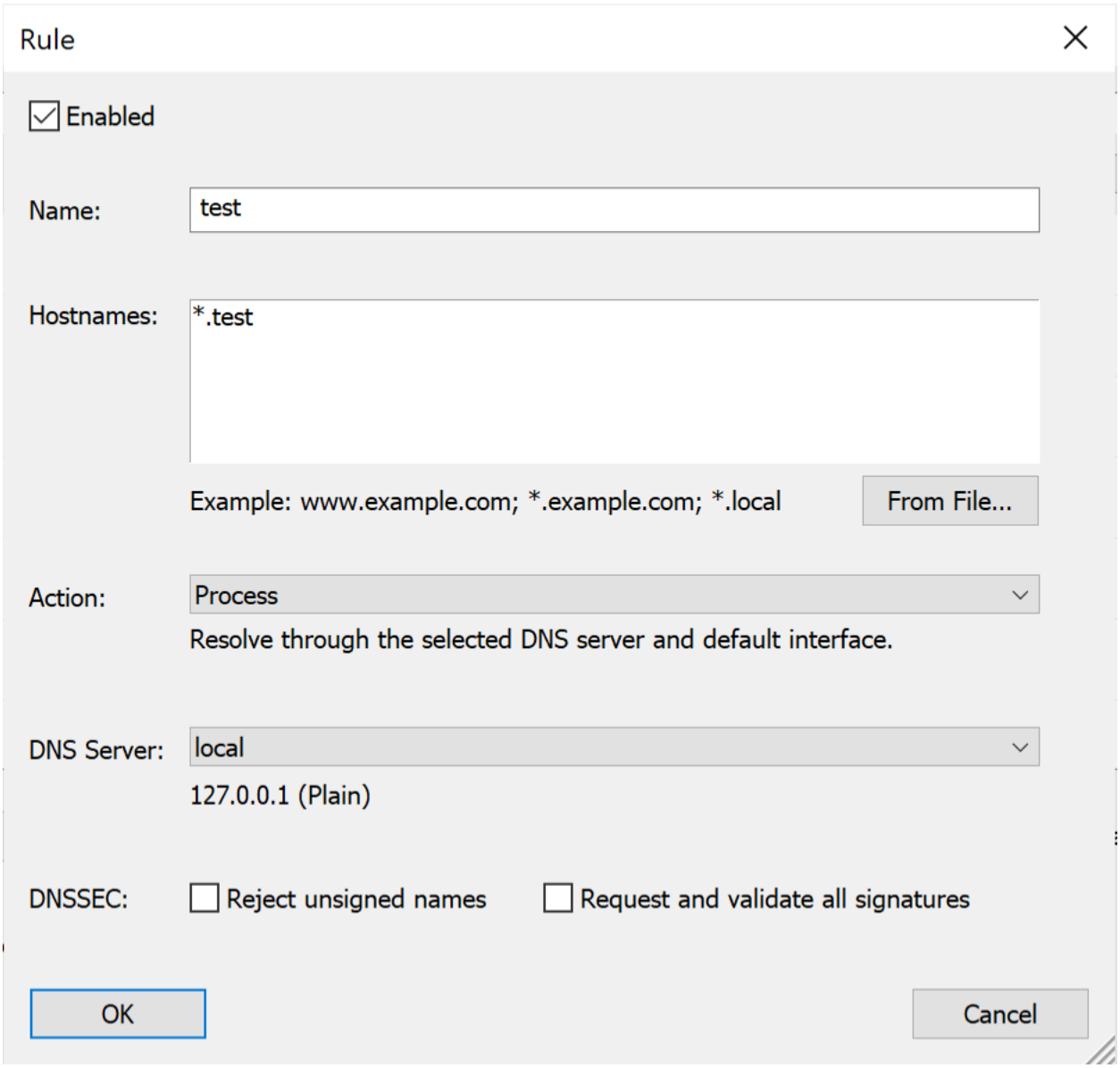
You will have to add a local DNS resolver and utilize dnsmasq to resolve the *.test domain.

Using Yoga DNS it's quite simple.

First add a local DNS server:



When you are done with the server configuration, you have to add a rule for *.test:



Mac

On macOS, DNS resolution is configured automatically for `*.test` domains using a feature macOS inherits from BSD. When `reward install` is run (or `reward svc up` for the first time) the following contents are placed in the `/etc/resolver/test` file. This has the effect of having zero impact on DNS queries except for those under the `.test` TLD.

```
nameserver 127.0.0.1
```

If you desire to have more than this route through the `dnsmasq` container, you could place another similar file in the `/etc/resolver/` directory on a per-TLD basis, or alternatively configure all DNS lookups to pass through the `dnsmasq` container. To do this, open up Advanced connection settings for the Wi-Fi/LAN settings in System Preferences, and go to the DNS tab. In here press the "+" button to add a new DNS record with the following IP address: `127.0.0.1` followed by fallback records:

```
127.0.0.1
1.1.1.1
```

Bind dnsmasq container to both TCP and UDP ports

By default, only the UDP port 53 is exposed from the dnsmasq container. Sometimes it doesn't seem to be enough, and the TCP port 53 has to be exposed as well. To do so enable the `reward_dnsmasq_bind_tcp` variable in the ~/.reward.yml file.

```
reward_dnsmasq_bind_tcp: true
reward_dnsmasq_bind_udp: true
```

And restart the Reward services.

Ubuntu

Per network

Open up your connection (Wi-Fi/LAN) settings, and go to the IPv4 tab. Turn off the automatic DNS setting, and enter the following IP addresses

```
127.0.0.1, 1.1.1.1
```

Persistent global configuration

To avoid having to set the DNS servers for each network you connect, you can also choose to update the global DNS configuration.

Use the `resolvconf` service to add a permanent entry in your `/etc/resolv.conf` file.

Install resolvconf

```
sudo apt update && sudo apt install resolvconf
```

Edit the `/etc/resolvconf/resolv.conf.d/base` file as follows:

```
search home net
nameserver 127.0.0.1
nameserver 1.1.1.1
```

Restart network-manager

```
sudo service network-manager restart
```

Note: In the above examples you can replace `1.1.1.1` (CloudFlare) with the IP of your own preferred DNS resolution service such as `8.8.8.8` and `8.8.4.4` (Google) or `9.9.9.9` and `149.112.112.112` (Quad9)

Fedora

Per network

Open up your connection (Wi-Fi/LAN) settings, and go to the IPv4 tab. Turn off the automatic DNS setting, and enter the following IP addresses

```
127.0.0.1, 1.1.1.1
```

Persistent global configuration

You should enable systemd-resolved and change your resolv.conf to be managed by systemd-resolved.

```
$ sudo systemctl start systemd-resolved
$ sudo systemctl enable systemd-resolved
$ sudo ln -sf ln -s /run/systemd/resolve/resolv.conf /etc/resolv.conf
```

When your systemd-resolved is ready, run reinstall reward's DNS settings and restart systemd-resolved.

```
$ reward install --dns
$ sudo systemctl restart systemd-resolved
```

DNS resolution to Traefik inside docker network

By default, inside the environment's docker network the environment's hostname will be resolved to the traefik container's IP address.

To change this behaviour you can disable it using the following setting in `~/.reward.yml`

```
reward_resolve_domain_to_traefik: false
```

1.6.4 Blackfire Profiling

For information on what Blackfire is, please see the introduction to Blackfire in Blackfire documentation.

Blackfire may be enabled on both `magento1` and `magento2` env types by adding the following to the project's `.env` file (or exporting them to environment variables prior to starting the environment):

```
REWARD_BLACKFIRE=true

BLACKFIRE_CLIENT_ID=<client_id>
BLACKFIRE_CLIENT_TOKEN=<client_token>
BLACKFIRE_SERVER_ID=<server_id>
BLACKFIRE_SERVER_TOKEN=<server_token>
```

Note: You can obtain the IDs and Tokens used in the above from within your Blackfire account under Account Settings -> Credentials or from the credentials are of the environment you're pushing profile information into.

CLI Tool

To use the Blackfire CLI Tool, you can run `reward blackfire [arguments]`.

For more information on the CLI tool, please see Profiling CLI Commands in Blackfire's documentation.

1.6.5 Docker Desktop on Linux

Background

If you are using Docker Desktop on Linux your user is not going to utilize docker-engine on your system. Instead, in the background Docker Desktop creates a Virtual Machine with Docker installed on it. After that Docker Desktop forwards the socket of this machine's docker to your machine in your home (eg: `~/.docker/desktop/docker.sock`) and sets your `docker` command (using docker contexts) to use this socket.

You can check these contexts using `docker context ls`.

But Reward still tries to use the default docker socket (`/run/docker.sock`).

Solution

To change this behaviour **permanently**, you can open `~/.reward.yml` and add the following line:

```
docker_host: /home/_YOUR_USER_/.docker/desktop/docker.sock
```

TIP: If you want to change the docker socket temporary you can set an environment variable before calling `reward`.

```
DOCKER_HOST=/home/_YOUR_USER_/.docker/desktop/docker.sock reward env ps
```

1.6.6 Expose ports from environment to the host machine

It's possible to expose ports from the environment to the host machine. This is useful when you want to access the environment from the host machine (eg. you run an application directly on the host machine and you want to access the Magento database.)

To expose ports from the environment to the host machine, add the following line to the project's `.env` file.

```
MYSQL_EXPOSE=true
REDIS_EXPOSE=true
OPENSEARCH_EXPOSE=true
ELASTICSEARCH_EXPOSE=true
RABBITMQ_EXPOSE=true
```

When it's done, restart the environment.

```
reward env down
reward env up
```

Please note that you cannot expose the same service twice. Eg. if you have `MYSQL_EXPOSE=true` in one environment and you want to expose mysql from another environment that would cause a port conflict. You have to select a different port using `MYSQL_EXPOSE_TARGET`.

The same applies for the rest of the services.

The default ports:

```
MYSQL_EXPOSE_TARGET=3306
REDIS_EXPOSE_TARGET=6379
OPENSEARCH_EXPOSE_TARGET=9200
ELASTICSEARCH_EXPOSE_TARGET=9200
RABBITMQ_EXPOSE_TARGET=5672
```

Using this method you will be able to reach the database of one environment from another environment.

To do so, you can use `host.docker.internal` as the mysql host name in the secondary container.

1.6.7 Expose Reward HTTP/HTTPS ports to the local network

It's possible to expose Reward HTTP/HTTPS ports to the local network. This is useful when you want to access the environment from the local network.

To expose the HTTP and HTTPS ports of traefik to the local network, add the following line to `~/.reward.yml`:

```
reward_traefik_listen: 0.0.0.0
```

When it's done, restart Traefik.

```
reward svc down
reward svc up
```

1.6.8 LiveReload Setup

LiveReload routing is currently supported on the `magento2`, `pwa-studio` and `shopware` environment types. Other environment types may utilize LiveReload via per-project compose configurations to set up the routing for LiveReload JS and WebSocket endpoints.

Configuration for Magento 2

Magento 2 bundles an example grunt based server-side compilation workflow which includes LiveReload, and it works within the Reward shell environment. In order to use this:

1. Rename or copy `Gruntfile.js.sample` file to `Gruntfile.js` in your project root.
2. Rename or copy `package.json.sample` file to `package.json` in your project root.
3. Run `npm install` to install the required NodeJS packages as defined in `package.json`.
4. Merge the following into your project's `app/etc/env.php` configuration file:

```
<?php
return [
    'system' => [
        'default' => [
            'design' => [
                'footer' => [
                    'absolute_footer' => '<script defer src="/livereload.js?port=443"></
↪script>'
                ]
            ]
```

(continues on next page)

(continued from previous page)

```
        ]
    ]
];
```

Note: This can be accomplished via alternative means, the important part is the browser requesting `/livereload.js?port=443` when running the site on your local development environment.

1. Run `bin/magento app:config:import` to load merged configuration into the application and flush the cache `bin/magento cache:flush`.

With the above configuration in place, you'll first enter the FPM container via `reward shell` and then setup as follows:

1. Clean and build the project theme using grunt:

```
$ grunt clean
$ grunt exec:blank
$ grunt less:blank
```

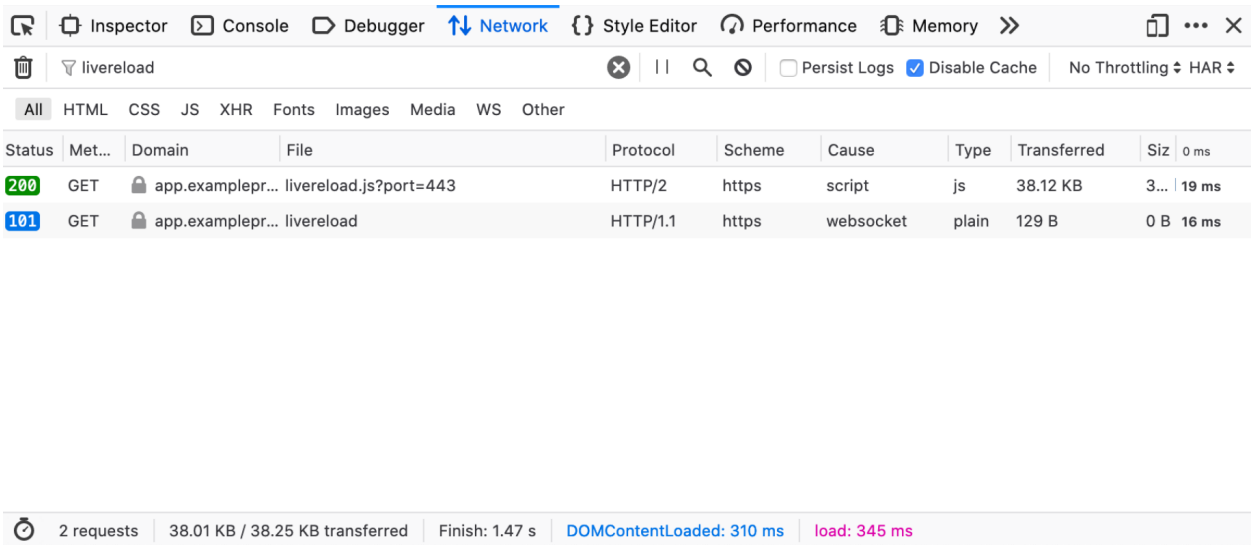
1. Thereafter, only a single command should be needed for daily development:

```
$ grunt watch
```

Note: Grunt should be used within the php-fpm container entered via `reward shell`

This setup will also be used to persist changes to your compiled CSS. When you run `grunt watch`, a LiveReload server will be started on ports 35729 within the php-fpm container and Traefik will take care of proxying the JavaScript tag and WebSocket requests to this listener.

On a working setup with `grunt watch` running within `reward shell` you should see something like the following in the network inspector after reloading the project in a web browser.



Configuration for Shopware Development Template

Shopware Hot Reload functionality requires to open additional ports on the same hostname and disable https redirects.

To configure the additional ports, you will have to update Traefik configuration.

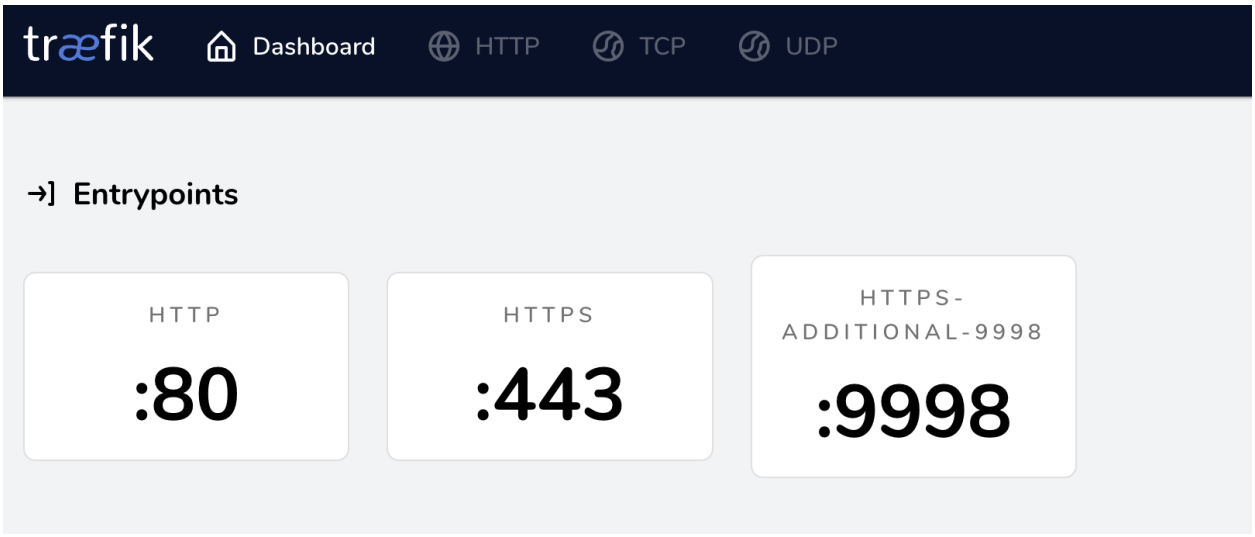
Open Reward Global Configuration (default: `~/.reward.yml`) and add the following line:

```
reward_traefik_bind_additional_https_ports: [ 9998, 9999 ]
reward_traefik_allow_http: true
```

When it's done, restart Traefik.

```
reward svc down
reward svc up
```

If you open Traefik dashboard, you should see the new ports in the entrypoints section.



When it's done you have to configure this additional port on the PHP container as well. Also, make sure to enable adding the required headers by Traefik.

Add the following line to the environment's `.env` file:

```
REWARD_HTTPS_PROXY_PORTS=9998,9999
REWARD_TRAEFIK_CUSTOM_HEADERS=hot-reload-mode=1,hot-reload-port=9999
```

Warning: If you want to disable hot-reload-mode ensure to remove (or comment out) the *REWARD_TRAEFIK_CUSTOM_HEADERS* line from the *.env* file.

And restart the environment.

```
reward env down
reward env up
```

Now if you start the Live Reload server, the requests will be proxied to it.

```
reward shell
./psh.phar storefront:hot-proxy
```

Configuration for Shopware Production Template

Similarly to the development template, you have to configure Traefik to allow additional ports and disable https redirects.

Open Reward Global Configuration (default: `~/.reward.yml`) and add the following line:

```
reward_traefik_bind_additional_https_ports: [ 9998, 9999 ]
reward_traefik_allow_http: true
```

When it's done, restart Traefik.

```
reward svc down
reward svc up
```

Open the environment's `.env` file and add the following line:

```
REWARD_HTTPS_PROXY_PORTS=9998,9999
REWARD_TRAEFIK_CUSTOM_HEADERS=hot-reload-mode=1,hot-reload-port=9999
```

> **Warning:** If you want to disable hot-reload-mode ensure to remove (or comment out) the *REWARD_TRAEFIK_CUSTOM_HEADERS* line from the *.env* file.

And restart the environment.

```
reward env down
reward env up
```

Now if you start the Live Reload server, the requests will be proxied to it.

```
reward shell
bin/build-storefront.sh
bin/watch-storefront.sh
```

1.6.9 Magento 2 Testing

Magento 2 Testing

To enable testing components, set the following configuration in your project's `.env` file:

```
REWARD_TEST_DB=true
```

This will launch an additional MySQL 5.7 database instance running on `tempfs` (blazing-fast memory) storage.

Temporary file locations have also been mounted into `tempfs` memory storage:

- `/var/www/html/dev/tests/integration/tmp/`

Running Unit Tests

Create your `phpunit.xml` using contents of `phpunit.xml.dist`. We recommend customizing values of:

- Memory usage `TESTS_MEM_USAGE_LIMIT` with value of `8G` should be enough

```
<php>
    <const name="TESTS_MEM_USAGE_LIMIT" value="8G"/>
</php>
```

That's it! Now you are ready to run Unit Tests.

Execution

- To run all tests declared in `phpunit.xml` execute: `vendor/bin/phpunit -c dev/tests/unit/phpunit.xml`
- If you need to run only specific directory, execute: `vendor/bin/phpunit -c dev/tests/unit/phpunit.xml {PATH TO TESTS}`

Debugging

If you have configured Xdebug, run Unit tests inside **Debug** console (`reward debug` instead of `reward shell`). The code execution will stop at the breakpoints.

Running Javascript Unit Tests

1. Configure your `.env` and set `NODE_VERSION=16`
2. Launch a shell session within the project environment's `php-fpm` container with `reward shell`
3. Install javascript unit test dependencies with `npm install`
4. Deploy static content with

```
bin/magento setup:static-content:deploy -f
```

Execution

```
$ grunt spec:<THEME>
```

For more specific jasmine unit test instructions, see the Magento DevDocs (Magento 2.4)

Troubleshooting

- You must be within your project environment's `php-fpm` container before running `npm install`. If you are having issues installing node packages, remove your `node_modules` directory with `rm -rf node_modules/ package-lock.json` and then retry `npm install`.
- If you have an issue with `jasmine` tests being unable to execute it might be due to installing the wrong versions of node `grunt-contrib-jasmine`. You can fix this by using:

```
cp package.json.sample package.json && rm -rf node_modules/ package-lock.json && npm␣
↪install
```

Running Integration Tests

All the necessary files are located in `dev/tests/integration/`:

1. Create your `phpunit.xml` using contents of `phpunit.xml.dist`. We recommend customizing values of:
   - Maximum memory usage `TESTS_MEM_USAGE_LIMIT` with value of `8G` should be enough
   - Magento deployment mode `TESTS_MAGENTO_MODE` should be covered both for `developer` and `production`
   - Significantly increase the speed with `TESTS_PARALLEL_RUN` set to 1
2. You need to create `etc/install-config-mysql.php` based on `etc/install-config-mysql.php.dist` as a template. The arguments are exactly the same to those you use for `bin/magento setup:install`:

```
return [
    'db-host' => 'tmp-mysql',
    'db-user' => 'root',
    'db-password' => 'magento',
    'db-name' => 'magento_integration_tests',
    'backend-frontname' => 'backend',
    'search-engine' => 'elasticsearch7',
    'elasticsearch-host' => 'elasticsearch',
    'elasticsearch-port' => 9200,
    'admin-user' => \Magento\TestFramework\Bootstrap::ADMIN_NAME,
    'admin-password' => \Magento\TestFramework\Bootstrap::ADMIN_PASSWORD,
    'admin-email' => \Magento\TestFramework\Bootstrap::ADMIN_EMAIL,
    'admin-firstname' => \Magento\TestFramework\Bootstrap::ADMIN_FIRSTNAME,
    'admin-lastname' => \Magento\TestFramework\Bootstrap::ADMIN_LASTNAME,
    'amqp-host' => 'rabbitmq',
    'amqp-port' => '5672',
    'amqp-user' => 'guest',
    'amqp-password' => 'guest',
    'session-save' => 'redis',
    'session-save-redis-host' => 'redis',
    'session-save-redis-port' => 6379,
    'session-save-redis-db' => 2,
    'session-save-redis-max-concurrency' => 20,
    'cache-backend' => 'redis',
    'cache-backend-redis-server' => 'redis',
    'cache-backend-redis-db' => 0,
    'cache-backend-redis-port' => 6379,
```

(continues on next page)

(continued from previous page)

```
    'page-cache' => 'redis',
    'page-cache-redis-server' => 'redis',
    'page-cache-redis-db' => 1,
    'page-cache-redis-port' => 6379,
];
```

3. You need to create `etc/config-global.php` based on `config-global.php.dist`. This is your container for Config data
    - for example: Configuration of Elasticsearch connection!

```
return [
    'customer/password/limit_password_reset_requests_method' => 0,
    'admin/security/admin_account_sharing' => 1,
    'admin/security/limit_password_reset_requests_method' => 0
];
```

That's it! Now you are ready to run your first Integration Tests.

Execution

There's one thing you should be aware of: **always provide full path to `phpunit.xml`**.

- To run all tests declared in `phpunit.xml` execute: `vendor/bin/phpunit -c $(pwd)/dev/tests/integration/phpunit.xml`
- If you need to run only specific directory, execute: `vendor/bin/phpunit -c $(pwd)/dev/tests/integration/phpunit.xml {ABSOLUTE PATH TO TESTS}`

Debugging

If you have configured Xdebug, run Integration tests inside **Debug** console (`reward debug` instead of `reward shell`). The code execution will stop at the breakpoints.

Troubleshooting

- In case you're getting a message like `Fatal error: Allowed memory size of ...` try to add prefix `php -dmemory_limit=-1` to your command, like

```
php -d memory_limit=-1 vendor/bin/phpunit -c $(pwd)/dev/tests/integration/phpunit.
↪xml
```

- If you're getting a message like `The store that was requested wasn't found. Verify the store and try again.` - run the following command

```
rm -Rf app/etc/env.php app/etc/config.php dev/tests/integration/tmp/*
```

Running Setup Integration Tests

All the necessary files are located in `dev/tests/setup-integration/`:

1. Create your `phpunit.xml` using contents of `phpunit.xml.dist`. We recommend customizing values of:
   - Install config file `TESTS_INSTALL_CONFIG_FILE` should be `etc/install-config-mysql.php`
   - Tests cleanup `TESTS_CLEANUP` should be set to `enabled`
   - Magento deployment mode `TESTS_MAGENTO_MODE` should be covered both for `developer` and `production` (set to `developer` for start)
2. You need to create `etc/install-config-mysql.php` based on `etc/install-config-mysql.php.dist` as a template. Example:

```
return [
   'default' => [
       'db-host' => 'tmp-mysql',
       'db-user' => 'root',
       'db-password' => 'magento',
       'db-name' => 'magento_integration_tests',
       'db-prefix' => '',
       'backend-frontname' => 'admin',
       'admin-user' => 'admin',
       'admin-password' => '123123q',
       'admin-email' => \Magento\TestFramework\Bootstrap::ADMIN_EMAIL,
       'admin-firstname' => \Magento\TestFramework\Bootstrap::ADMIN_FIRSTNAME,
       'admin-lastname' => \Magento\TestFramework\Bootstrap::ADMIN_LASTNAME,
       'enable-modules' => 'Magento_TestSetupModule2,Magento_TestSetupModule1,
↪Magento_Backend',
       'disable-modules' => 'all'
   ],
   'checkout' => [
       'host' => 'tmp-mysql',
       'username' => 'root',
       'password' => 'magento',
       'dbname' => 'magento_integration_tests'
   ],
   'sales' => [
       'host' => 'tmp-mysql',
       'username' => 'root',
       'password' => 'magento',
       'dbname' => 'magento_integration_tests'
   ]
];
```

That's it! Now you are ready to run your first Setup Integration Tests.

Execution

There's one thing you should be aware of: **always provide full path to `phpunit.xml`**.

- To run all tests declared in `phpunit.xml` execute: `vendor/bin/phpunit -c $(pwd)/dev/tests/setup-integration/phpunit.xml`
- If you need to run only specific directory, execute: `vendor/bin/phpunit -c $(pwd)/dev/tests/setup-integration/phpunit.xml {ABSOLUTE PATH TO TESTS}`

Debugging

If you have configured Xdebug, run Integration tests inside **Debug** console (`reward debug` instead of `reward shell`). The code execution will stop at the breakpoints.

Running API Functional Tests

All the necessary files are located in `dev/tests/api-functional/`.

1. Create your own `phpunit_{type}.xml` file using contents of `phpunit_{type}.xml.dist`. You **need** to configure:
   - Magento installation URL (with protocol) `TESTS_BASE_URL` - for example `https://app.magento2.test/`
   - Admin credentials `TESTS_WEBSERVICE_USER` and `TESTS_WEBSERVICE_APIKEY` (it's formally **password**) *The Admin account should exist, it will be created only if `TESTS_MAGENTO_INSTALLATION` is enabled*
2. Configure your Magento Installation using `etc/install-config-mysql.php.dist` as a template. The arguments are exactly the same to those you use for `bin/magento setup:install`:

```
return [
    'db-host' => 'tmp-mysql',
    'db-user' => 'root',
    'db-password' => 'magento',
    'db-name' => 'magento_integration_tests',
    'cleanup-database' => true,

    'amqp-host' => 'rabbitmq',
    'amqp-port' => '5672',
    'amqp-user' => 'guest',
    'amqp-password' => 'guest',
    'session-save' => 'redis',
    'session-save-redis-host' => 'redis',
    'session-save-redis-port' => 6379,
    'session-save-redis-db' => 2,
    'session-save-redis-max-concurrency' => 20,
    'cache-backend' => 'redis',
    'cache-backend-redis-server' => 'redis',
    'cache-backend-redis-db' => 0,
    'cache-backend-redis-port' => 6379,
    'page-cache' => 'redis',
    'page-cache-redis-server' => 'redis',
    'page-cache-redis-db' => 1,
```

(continues on next page)

(continued from previous page)

```
    'page-cache-redis-port' => 6379,

    'language' => 'en_US',
    'timezone' => 'America/Los_Angeles',
    'currency' => 'USD',
    'backend-frontname' => 'backend',
    'base-url' => 'https://app.magento2.test/',
    'use-secure' => '1',
    'use-rewrites' => '1',
    'admin-lastname' => 'Admin',
    'admin-firstname' => 'Admin',
    'admin-email' => 'admin@example.com',
    'admin-user' => 'admin',
    'admin-password' => '123123q',
    'admin-use-security-key' => '0',
    'sales-order-increment-prefix' => time(),
];
```

3. You need to create `etc/config-global.php` based on `config-global.php.dist`. This is your container for Config data
    - for example: Configuration of Elasticsearch connection!

```
return [
    'catalog/search/engine' => 'elasticsearch6',
    'catalog/search/elasticsearch6_server_hostname' => 'elasticsearch',
];
```

Execution

There's one thing you should be aware of: **always provide full path to `phpunit.xml`**.

- To run all tests declared in `phpunit_{type}.xml` execute: `vendor/bin/phpunit -c $(pwd)/dev/tests/api-functional/phpunit_{type}.xml`
- If you need to run only specific directory, execute: `vendor/bin/phpunit -c $(pwd)/dev/tests/api-functional/phpunit_{type}.xml {ABSOLUTE PATH TO TESTS}`

Debugging

When debugging APIs you may need to use Xdebug - configure your `phpunit_{type}.xml`:

- `TESTS_XDEBUG_ENABLED` to `true`
- `TESTS_XDEBUG_SESSION` to `phpstorm`

Running MFTF Tests

All the MFTF-related operations are operated by `vendor/bin/mftf`, necessary files are located in `dev/tests/acceptance/`.

To run Acceptance tests you need to configure the MFTF environment. Once you've done that, follow these steps to run the tests.

1. Make sure that you enabled following in your `.env` file:
   - `REWARD_SELENIUM` - Responsible for running virtual browser for your tests
   - `REWARD_ALLURE` - Responsible for test results reporting
   - `REWARD_SELENIUM_DEBUG` - Enables you to preview the tests with VNC
2. Run `vendor/bin/mftf build:project`, the configuration files will be generated in `dev/tests/acceptance`.
3. Adjust `dev/tests/acceptance/.env` file by setting:
   - `MAGENTO_BASE_URL`
   - `MAGENTO_BACKEND_NAME` to your Backend path (Check with `bin/magento info:adminuri`)
   - `MAGENTO_ADMIN_USERNAME`
   - `MAGENTO_ADMIN_PASSWORD`
   - `SELENIUM_HOST` (by default it is `selenium`)

   Sample configuration

```
MAGENTO_BASE_URL=https://app.magento2.test/
MAGENTO_BACKEND_NAME=backend
MAGENTO_ADMIN_USERNAME=admin
MAGENTO_ADMIN_PASSWORD=123123q
BROWSER=chrome
MODULE_WHITELIST=Magento_Framework,ConfigurableProductWishlist,
↪ConfigurableProductCatalogSearch
ELASTICSEARCH_VERSION=7
SELENIUM_HOST=selenium
```

   More details can be found in Magento DevDocs.

Execution

- Execute single test`vendor/bin/mftf run:test -r AdminLoginTest`
- Execute group/suite of tests`vendor/bin/mftf run:group -r customer`

Debugging

For more information about Debugging MFTF - please follow the Magento Functional Testing Framework section. The process of debugging is based on VNC connection to the Chrome instance.

Magento 2 Functional Testing Framework

For information what **Magento Functional Testing Framework** is - please follow to MFTF DevDocs.

MFTF is part of Magento 2. To run tests you need Selenium instance with Chrome Webdriver. Reward provides Docker setup that contains Selenium Standalone with Chrome. You can enable it by adding the following to the project's `.env` file (or exporting them to environment variables prior to starting the environment):

```
REWARD_SELENIUM=true
```

After generating MFTF configuration files (`dev/tests/acceptance/.env` generated by `vendor/bin/mftf setup:env` command) , you need to provide selenium hostname:

```
SELENIUM_HOST=selenium
BROWSER=chrome
```

Running Tests

We provide complex instruction on How to run MFTF Tests in Reward environment.

Debugging MFTF Tests

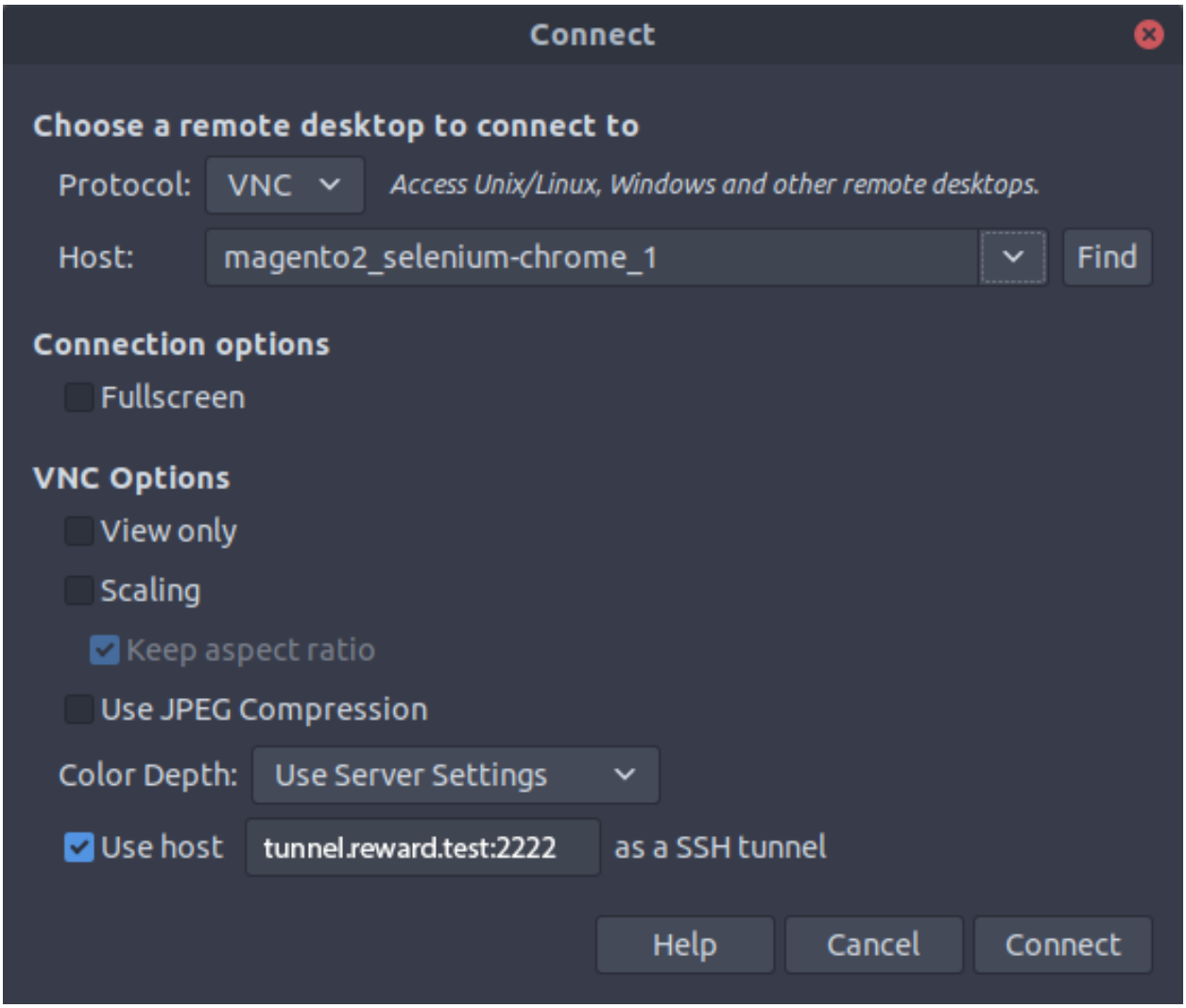
By default, Reward uses headless Chrome browser. If you want to preview the tests - you need to extend `.env` file and update environment containers (`reward env up -d`)

```
REWARD_SELENIUM_DEBUG=true
```

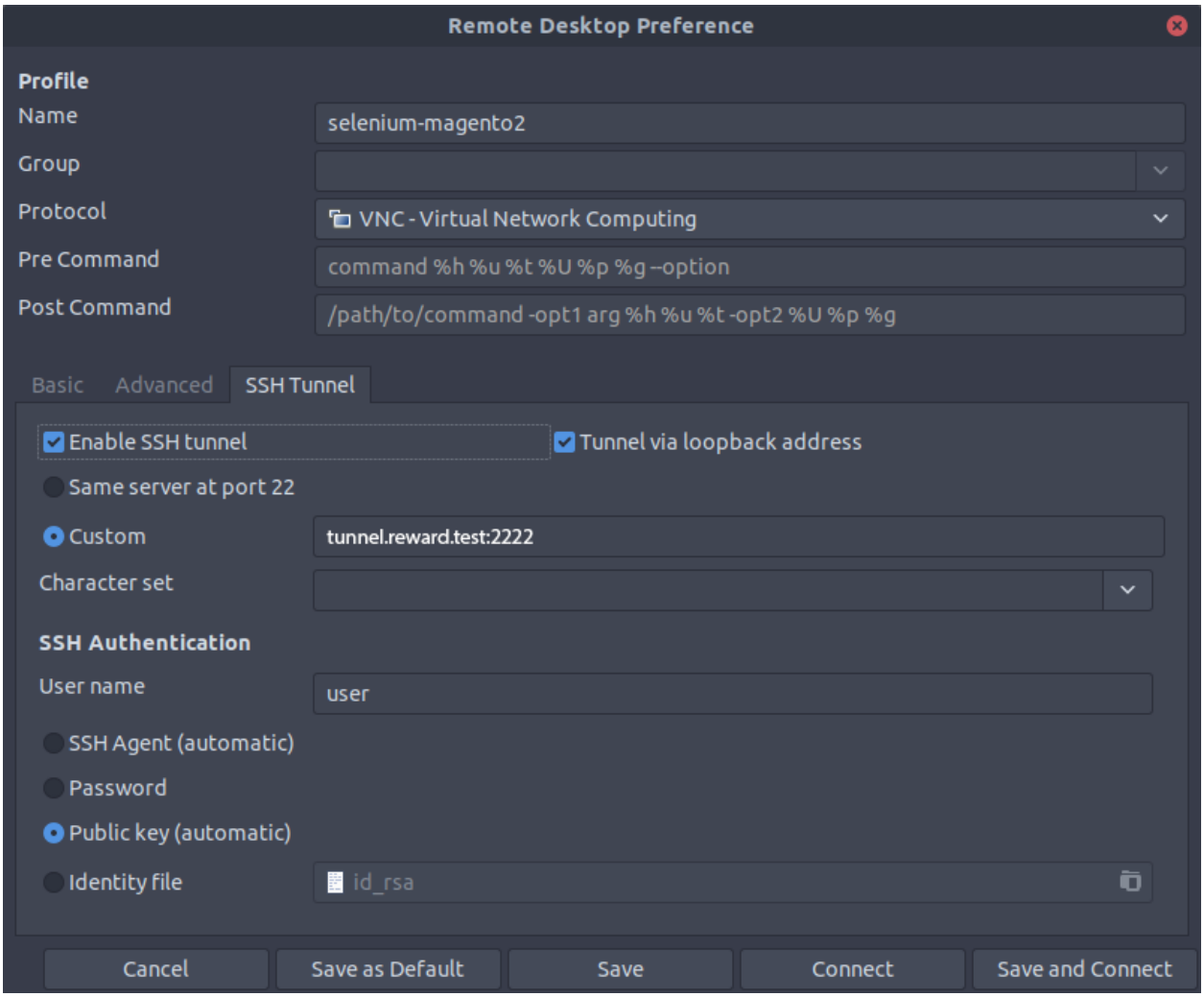
Default password for VNC session is `secret`

To preview the process of testing, you need any **VLC** client that provides **SSH Tunnel** support (e.g. Remmina). To preview the process of testing, you need to use `tunnel.reward.test:2222` (login: `user`):

Remote Desktop Viewer



Remmina



Mac OS X

To preview the process in Mac OS X, you must first create an SSH tunnel to the docker instance hosting the VNC server. That would look something like:

```
ssh -N -L localhost:5901:magento2_selenium_1:5900 tunnel.reward.test
```

Where `5901` is the port on your local computer you want to use to access the VNC server. Then, using Finder you can " Go > Connect to Server" `vnc://localhost:5901`.

1.6.10 Open additional shared http/https port(s)

It's possible to configure additional shared HTTP/s port(s) and proxy the requests to the PHP containers.

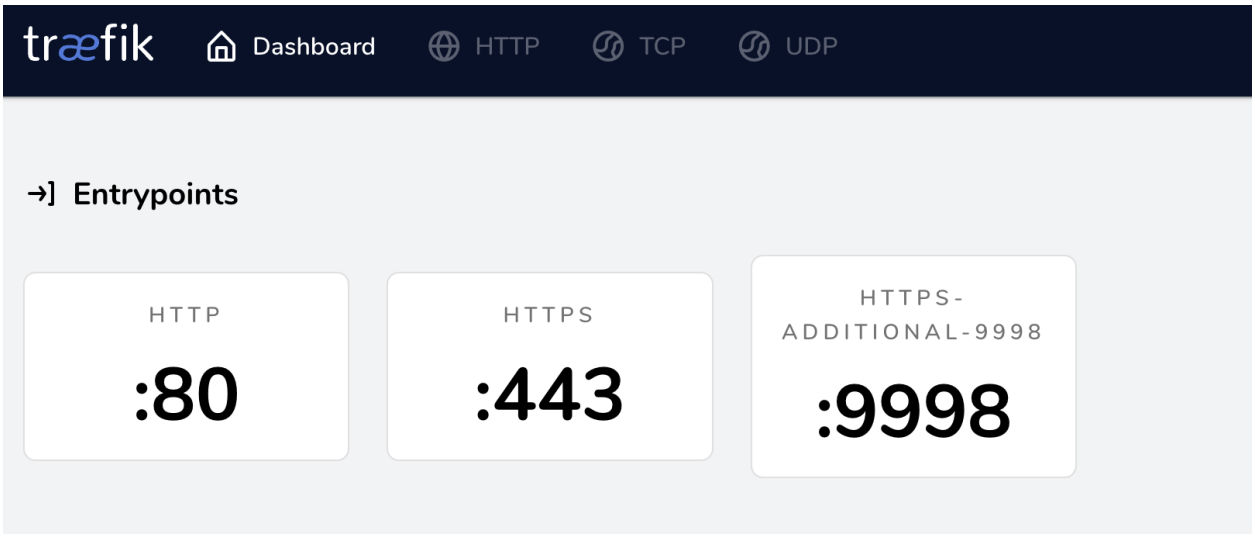
First, configure Traefik. Add the port to the Reward Configuration (default: `~/.reward.yml`).

```
reward_traefik_bind_additional_http_ports: [ 8080 ]
reward_traefik_bind_additional_https_ports: [ 8443, 9998 ]
```

When it's done, restart Traefik.

```
reward svc down
reward svc up
```

Open Traefik dashboard to check if the new port(s) exist in the entrypoints section.



When it's done configure this additional port on the PHP container as well. Add the following line to the project's `.env` file. (It's also possible to add multiple ports, separated by comma.)

```
REWARD_HTTP_PROXY_PORTS=8080
REWARD_HTTPS_PROXY_PORTS=8443,9998
```

And restart the environment.

```
reward env down
reward env up
```

1.6.11 SPX Support

To enable SPX support, you need to set the `REWARD_SPX=true` environment variable in the `.env` file.

Then run `reward env up` to apply the changes.

In this case a new container will be created, called `php-spx`, which will have the SPX extension installed.

All requests that contain one of the following will be forwarded to the `php-spx` container:

- `SPX_ENABLED` cookie
- `SPX_KEY` cookie

- SPX_ENABLED argument in the URL (e.g. https://m2.test/?SPX_ENABLED=1)
- SPX_KEY argument (e.g. ?SPX_KEY=dev)
- SPX_UI_URI argument (e.g. ?SPX_UI_URI=/)

Warning: If you **open the SPX UI**, the **SPX cookies will be automatically set** in your browser. If you want to disable sending all requests to the SPX container, you need to clear the cookies.

In similar fashion to the `reward shell` command there is also an spx command to launch into an SPX enabled container shell for debugging CLI workflows:

```
reward spx
```

In this container if you run any commands those will be executed with the SPX profiling enabled.

SPX UI

To access the SPX UI, you need to navigate to the environment URL with the SPX_UI_URI argument set to /.

E.g.: https://m2.test/?SPX_KEY=dev&SPX_UI_URI=/

If you run `reward info` you will see the SPX UI URL in the output.

SPX Key

The SPX_HTTP_KEY environnment variable can be set to a value of your choice. This key will be used to authenticate requests to the SPX container.

It defaults to be dev.

If you configure the SPX_HTTP_KEY environment variable, you will need to pass the key in the URL to access the SPX UI.

E.g.: https://m2.test/?SPX_KEY=dev&SPX_UI_URI=/

But `reward info` will show you the correct URL to access the SPX UI.

Configuration

Reference documentation for the SPX variables are available here.

The following variables can be configured using environment variables:

- SPX_DEBUG - default: 1
- SPX_HTTP_ENABLED - default: 1
- SPX_HTTP_KEY - default: "dev"
- SPX_HTTP_IP_WHITELIST - default: *
- SPX_HTTP_PROFILING_ENABLED - default: 1
- SPX_HTTP_PROFILING_AUTO_START - default: 1
- SPX_HTTP_TRUSTED_PROXIES - default: *

1.6.12 WSL and WSL2 Support

On Windows, by default, Reward uses file synchronization method with Mutagen instead of direct mounting. It is possible to enable direct mounting for WSL2. WSL2 provides better performance for Filesystem mounts if they are in the WSL filesystem.

Under the hood Windows runs a lightweight Virtual Machine and using Bash for Ubuntu for Windows you will get your root from this filesystem. If you choose to develop from under this filesystem you will get really great performance for your mounts.

The caveat is you'll have to reach this filesystem somehow from your IDE.

> **Warning:** It's pretty important to understand in this case the files you are reaching from WSL2 are inside a Virtual Machine.
>
> That's the reason why they are blazing fast in docker, because they can be directly mounted to docker just as on a Linux machine.

Further reading:

- Explanation on Architecture in WSL2
- Docker Desktop WSL 2 backend

Enable WSL 2 on Windows 10

1. To install WSL2, enable WSL and VirtualMachinePlatform Windows features in an **elevated command prompt**

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /
↪all /norestart

dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

2. Then install the new version of Microsoft's Linux kernel

   WSL Update

3. Change default WSL version of the system to WSL2

```
wsl --set-default-version 2
```

4. Install the preferred Linux Distribution from Microsoft Store
5. Check WSL versions of the distros on the system

```
wsl --list
```

6. Convert distributions to WSL 2

```
wsl --set-version <distribution name> <versionNumber>

# example:
wsl --set-version Ubuntu-20.04 2
```

Install WSL2 on Windows 10

Warning: **Beware of your Distro's package managers!** Please make sure you are using Windows' Docker's WSL integration.

You should not install Docker with Ubuntu's package manager inside your Bash for Ubuntu for Windows!

Using WSL2 with Reward

To use Reward with WSL2 there are two possible options:

1. Use Reward's Windows binary and enable **wsl2-direct-mount** option
2. Use Reward's Native Linux binary and trick Reward to think it's running on a Linux host

Using WSL2 Direct Mount option

To enable WSL2 support in Reward you can disable syncing with the following setting in your reward config `~/.reward.yml`.

```
reward_sync_enabled: false
```

Using Linux native binary

In this case you will just have to download and install the Linux installation method and install Reward as if you would do in a Linux machine.

Warning: If you choose to use the Reward linux binary on Windows the `reward install` command **will not install** the root CA certificate to your Windows Trusted Root CA Store.

To do this, you will have to import it manually.

- Start > Run > MMC
- File > Add / Remove Snap In
- Double Click on `Certificates`
- Select Computer Account
- Local Computer > Finish
- Open Trusted Root Certification Authorities > Certificates (Local Computer)
- Right Click on the Store > All tasks > Import
- Next, next and select the browse button
- On the appearing window's address bar type `\\wsl$`
- Enable showing of all filetypes
- Go to Ubuntu > home > username > .reward > ssl > rootca > certs
- Select the `ca.cert.pem` and import it

Reaching WSL2 filesystem in Windows

Open File Explorer and type `\\wsl$` to the address bar.

Visual Studio Code also offers a WSL2 extension for this problem.

- WSL 2 with Visual Studio Code

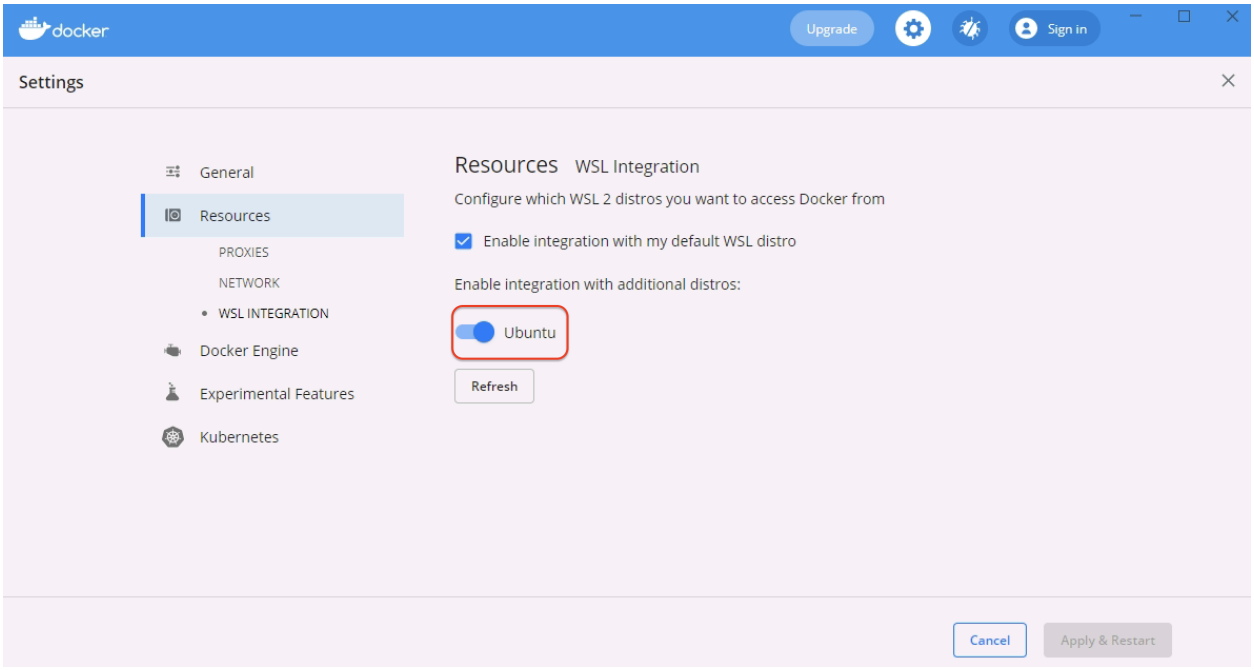
> **Warning:** You should only use this for directories which are inside your WSL2 filesystem or mounted to WSL2 filesystem.

- Get started mounting a Linux disk in WSL 2

Note: If you encounter the following error:

```
Cannot create container for service php-fpm:
failed with: can't access specified distro mount service:
stat /run/guest-services/distro-services/ubuntu.sock: no such file or directory
```

Possible Solution: check your Windows Docker Settings and under the Resources -> WSL INTEGRATION tab and make sure docker integration is enabled for the distro you are using.



Reward config file overlapping in WSL/Windows (only when you are using Windows binary)

Note: Even when you are running Reward's Windows binary inside WSL2 it will not use WSL2's home directory as your user's and the application's HOME. That's because of GO's behaviour on determining the User's HOME. On Windows it will still be your Windows User's HOME directory.

To change Reward's default config file you can use the `--config` flag.

Similarly Composer's home directory (~/.composer) will be mounted from Windows User's home directory.

To change it you can add `reward_composer_dir` variable to your Reward configuration and set a value from WSL's filesystem.

1.6.13 Xdebug Support

There are two docker containers running FPM, `php-fpm` and `php-debug`. The `php-debug` container has the Xdebug extension pre-installed. Nginx will automatically route requests to the `php-debug` container when the `XDEBUG_SESSION` cookie has been set to `PHPSTORM` via the Xdebug Helper browser extension.

Xdebug will automatically connect back to the host machine on port `9000` for Xdebug2 and `9003` for Xdebug3 for each request routed to the `php-debug` container (i.e. when the `XDEBUG_SESSION` cookie is set). When configuring Xdebug Helper in your browser, make sure it is setting this cookie with the value `PHPSTORM`.

If you use a firewall, allow connection to port `9000` for Xdebug2 and `9003` for Xdebug3.

In similar fashion to the `reward shell` command there is also a debug command to launch into an xdebug enabled container shell for debugging CLI workflows:

```
reward debug
```

Xdebug Version

Reward supports both Xdebug 2 and Xdebug 3 (default).

If you'd like to use Xdebug version 2, you'll have to configure it explicitly. If it's empty or omitted, reward defaults to Xdebug 3.

Add the following line to your `.env` file:

```
XDEBUG_VERSION=2
```

VSCode

To configure a project in VSCode for debugging, add the following to `.vscode/launch.json` in the project directory:

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Listen for XDebug",
            "type": "php",
            "request": "launch",
```

(continues on next page)

(continued from previous page)

```
            // Change this to 9000 if you are using Xdebug2
            "port": 9003,
            "pathMappings": {
                "/var/www/html": "${workspaceRoot}"
            }
        }
    ]
}
```

Note: If your project has (for example) `REWARD_WEB_ROOT=/webroot` in it's `.env` file, to mount `webroot/` to `/var/www/html` rather than the top-level project directory, you may need to set the `pathMapping` above to `${workspaceRoot}/webroot` for the mapping to function correctly.

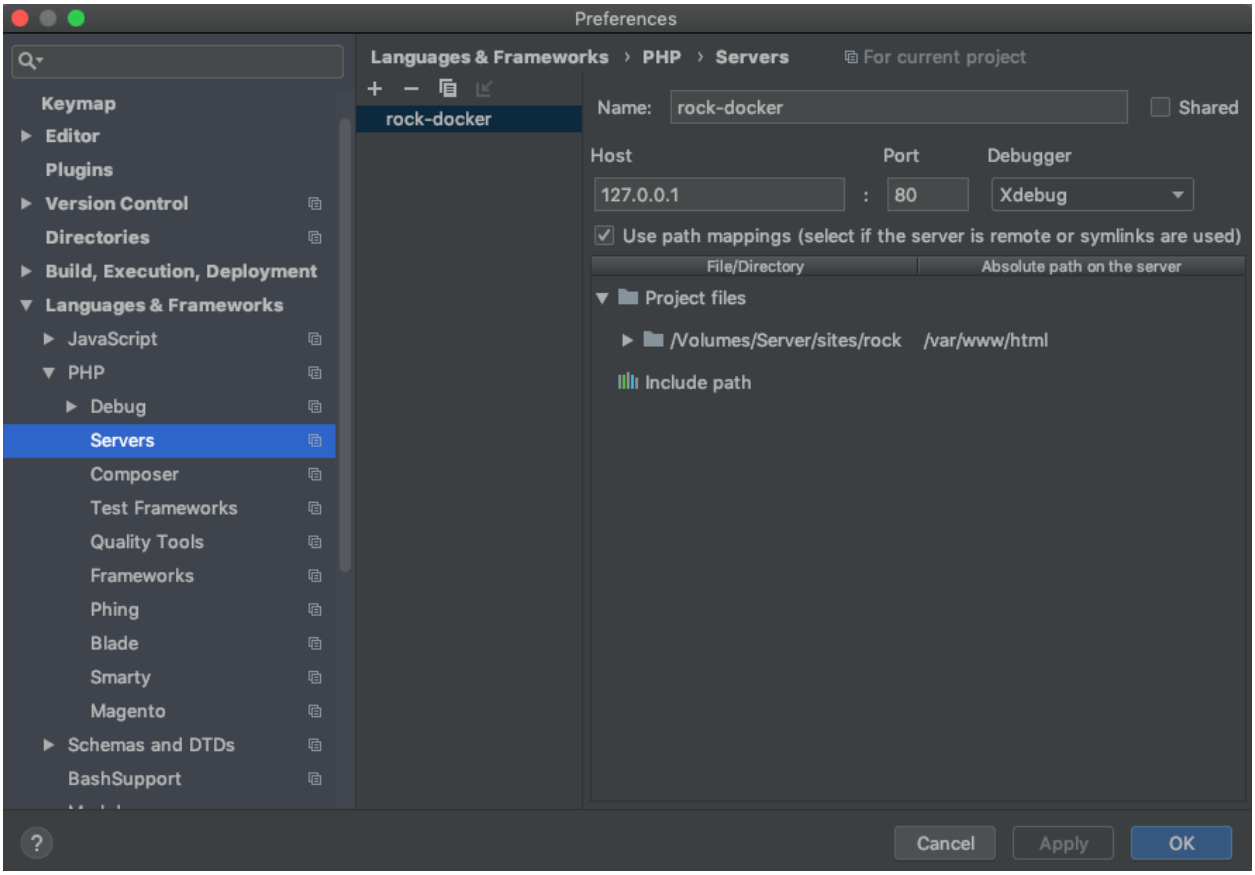
Once this configuration is in place, make sure you have the PHP Debug extension by Felix Becker installed. This is required for Xdebug support to function in VSCode. Additional information on launch settings specific to Xdebug use in VSCode may be found here.

To learn more about debugging in VSCode, please go here.

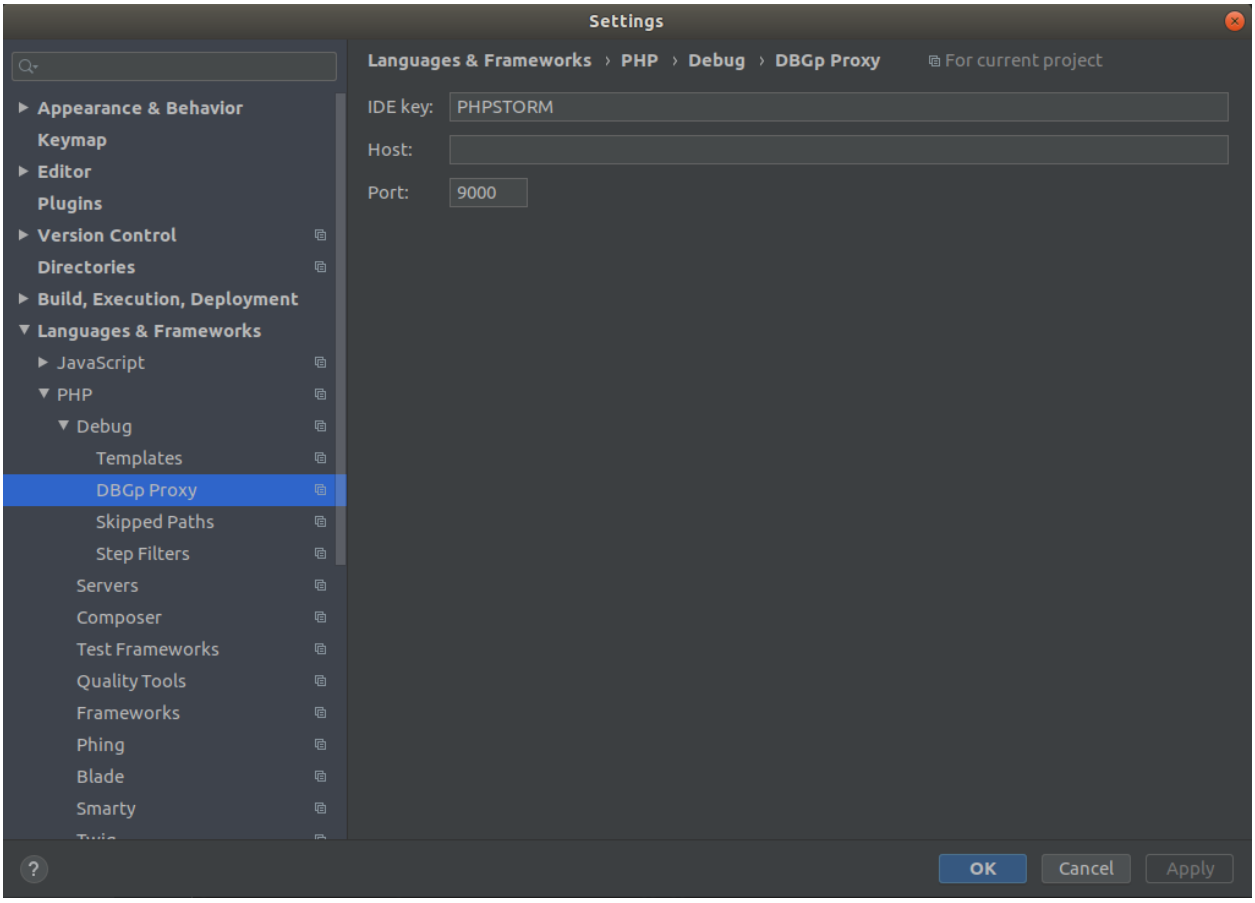
PhpStorm

When it receives the first request, PHP Storm should prompt you if the "Server" configuration is missing. The below image demonstrates how this is set up; the important settings are these:

- Name: `clnt-docker` (this is the value of the `REWARD_ENV_NAME` variable in the `.env` file appended with a `-docker` suffix)
- Host: `127.0.0.1`
- Port: `80`
- Debugger: Xdebug
- Use path mappings must be enabled, with a mapping to map the project root on the host to `/var/www/html` within the container.



Additional configurations may be required, such as configuring `DBGp Proxy` port.



Debugging PHP CLI Commands/Scripts

You can also debug PHP command line scripts using the following method.

```
# Open a shell in the environment debug container
reward debug

# Determine the Xdebug host address
php -i | grep -i 'xdebug.*host'

# The result was: 172.24.0.1

# Use the host address with the following example code to run bin/magento command.
# Xdebug2
php -dxdebug.remote_enable=1 -dxdebug.remote_mode=req -dxdebug.remote_port=9000 -dxdebug.
↪remote_host=172.24.0.1 -dxdebug.remote_connect_back=0 bin/magento

# Xdebug3
php -dxdebug.mode=debug -dxdebug.client_host=172.24.0.1 -dxdebug.client_port=9003 -
↪dxdebug.start_with_request=yes bin/magento
```

Source: Debug a PHP CLI script

Customizing Xdebug Remote Host (or Client Host)

By default, Xdebug is configured to connect to `host.docker.internal` that should resolve to the host machine IP address. For some reason in Ubuntu with classic Docker, this does not work. You can customize the remote host by setting the `XDEBUG_REMOTE_HOST` environment variable in the `.env` file.

To determine what address should be used, you can run the following command on the host machine:

```
# determine the ID or the name of the project's docker network
docker network ls

# the output should look like this
NETWORK ID     NAME                DRIVER    SCOPE
9632c97820b4   bridge              bridge    local
b32df7ba7038   host                host      local
233d483ba204   magento2            bridge    local
27dcd7a14e50   none                null      local
044c0e9d99cf   reward              bridge    local
```

In my case, I'd like to check the network of the project called `magento2`, so I'll need to use the name `magento2` or the ID `233d483ba204`.

```
docker network inspect magento2

# the output is a long json, the data we need is in the "IPAM.config.gateway" section
# it's enough to just show the first 20 lines of the output
docker network inspect magento2 | head -n 20

# or even better if we filter the output
docker network inspect --format "{{(index .IPAM.Config 0).Gateway}}" magento2
172.20.0.1
```

That should output an IP address, that can be added directly to the `.env` file.

```
XDEBUG_REMOTE_HOST=172.20.0.1
```

If that's ready, you can restart the php-debug container to apply the changes.

```
reward env up -- php-debug
```

Debugging Xdebug instance

```
reward debug
sudo bash
cd /etc/php/${PHP_VERSION}/mods-available
vim xdebug.ini
```

If you are not familiar with `vim`, you can use `nano` instead. Add the following line to the end of the configuration.

```
xdebug.remote_log=/proc/self/fd/2
```

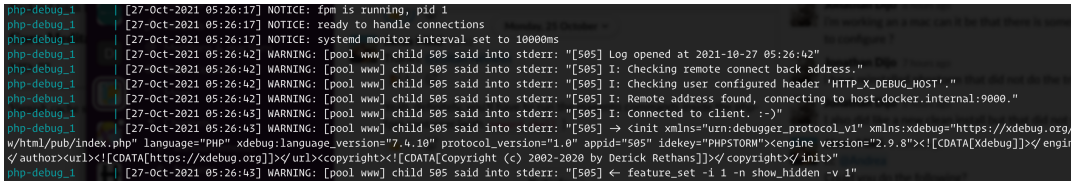
Now save the file and reload the php-fpm configuration using `kill -USR2`

```
kill -USR2 1
```

Exit the container and check its logs.

```
reward env logs -f php-debug
```

You should see something like this.



If you still cannot figure out what's wrong, you should check if you are able to connect to the Xdebug server from the container.

```
reward debug
sudo bash
apt-get update && apt-get install -y netcat

nc -v host.docker.internal 9003

# the output should be something like this
Connection to host.docker.internal 9003 port [tcp/*] succeeded!
```

```
# if you see something like this, then the connection is not working
nc: connect to host host.docker.internal port 9003 (tcp) failed: Connection refused/
↪timed out

# in this case a possible solution would be to update the firewall settings
# for example if you are using Ubuntu, you should enable the traffic from docker network␣
↪to the host
sudo ufw allow out on docker0 from 172.16.0.0/12
sudo ufw allow in on docker0 from 172.16.0.0/12
sudo ufw reload
```

```
# or if the hostname cannot be resolved then you'll see this
nc: getaddrinfo: Name or service not known

# in this case you should use the host machine's IP address instead of the hostname (see␣
↪the previous section)
```

1.7 Autocompletion

Reward fully supports autocompletion for the following shells:

- bash
- zsh
- fish
- powershell

1.7.1 Bash

```
$ source <(reward completion bash)

# To load completions for each session, execute once:
Linux:
$ reward completion bash > /etc/bash_completion.d/reward
MacOS:
$ reward completion bash > /usr/local/etc/bash_completion.d/reward
```

1.7.2 Zsh

If shell completion is not already enabled in your environment, you will need to enable it. You can execute the following once:

```
$ echo "autoload -U compinit; compinit" >> ~/.zshrc

# To load completions for each session, execute once:
$ reward completion zsh > "${fpath[1]}/_reward"

# You will need to start a new shell for this setup to take effect.
```

1.7.3 Fish

```
$ reward completion fish | source

# To load completions for each session, execute once:
$ reward completion fish > ~/.config/fish/completions/reward.fish
```

1.7.4 Powershell

```
PS> reward completion powershell | Out-String | Invoke-Expression

# To load completions for every new session, run:
PS> reward completion powershell > reward.ps1
# and source this file from your powershell profile.
```

1.8 FAQ

- Can I use on Windows?
  - Yes, Reward is cross-platform. It was written in Golang with intention to support Windows users as well.
- Should I run reward as **root** user or with sudo?
  - Nope, you should almost never use reward as root user or with sudo. The only exception is running the `reward self-update` command.
- Is Reward free?

  - Yes, and it's open source as well.
- Can I connect to the database using root user?
  - Yes, run `reward db connect --root`.

1.8.1 Frequent errors

- `docker api is unreachable`

  If you are sure Docker is running on your system, and you keep getting this error, you should check the following:

  - **Make sure your Docker version is up to date** and meets the system requirements mentioned in the Common Requirements section.

    Note: Package managers provide outdated Docker versions.

  - **Your user is not in the `docker` group**, or it cannot reach the docker socket.
    - After you add your user to the docker group ***you will have to reboot*** (or log out and log back in). For more info go to the following link: Install Docker Engine in Ubuntu See the `If you would like to use Docker as a non-root user` section.

    Note: You can check if your user is in the docker group with **id** command.

    You can make sure your user is able to reach the docker API running **docker ps** (without sudo).

- `Error: exit status x`

  Most of the cases these errors are coming from the container or docker itself. Reward tries to run a command inside the container, and the exit code of the command is not 0, or the container exited during the execution. In most cases this error code will be a part of a longer error message which will describe the problem.

- `Error: exit status 137`

  During Magento 2 installation (`reward bootstrap`) you will get this error code most likely during the `composer install` command. Composer installation needs a huge amount of memory and this error code represents a docker `out of memory` error code.

  To solve this problem, you will have to increase the memory limit of Docker Desktop. For more info see: Additional requirements (macOS only)

- `Error: unable to connect to beta: unable to connect to endpoint: unable to dial agent endpoint: unable to install agent: unable to get agent for platform: unable to locate agent bundle`

  This error message most probably appears if Mutagen is installed on your system but the Mutagen Agents Bundle are not. **Check your PATH (like `c:\bin`) and make sure `mutagen.exe` and `mutagen-agents.tar.gz` are there**. If not, download the latest release of Mutagen and extract the archive. You will find both files in it.

- `Error: create failed: unable to connect to beta: unable to connect to endpoint: unable to dial agent endpoint: unable to create agent command: unable to probe container: container probing failed under POSIX hypothesis (signal: killed) and Windows hypothesis (signal: killed)`

  If this error occurs, you can try to install mutagen-beta.

```
mutagen daemon stop
brew uninstall mutagen
brew install mutagen-io/mutagen/mutagen-beta
```

- `Error: unable to connect to daemon: client/daemon version mismatch (daemon restart recommended)`

  There's a possibility mutagen was updated on your system. Homebrew update doesn't restart mutagen daemon, try to run it manually:

```
mutagen daemon stop
mutagen daemon start
```

- `Package hirak/prestissimo has a PHP requirement incompatible with your PHP version, PHP extensions and Composer version`

  If you see this error message during the Magento 2 installation, you will have to downgrade your Composer version.

  To do so, add the following line to the `.env`:

```
COMPOSER_VERSION=1
```

  For more information, see the *Composer configuration*.

- `reward shell` stuck or frozen on Windows

  If any of Reward commands seems to be frozen or stuck, and you are using Git Bash on Windows, most probably you faced a known issue with Docker for Windows and Git Bash. To solve it, you have to use `winpty` (Windows Pseudo Terminal) using the following command:

```
winpty reward shell
```

  If you are interested in what's happening in the background, you can find more information here:

  - https://github.com/docker/for-win/issues/1588
  - https://willi.am/blog/2016/08/08/docker-for-windows-interactive-sessions-in-mintty-git-bash/

- Chrome of Firefox shows `NET::ERR_CERT_AUTHORITY_INVALID` or `SEC_ERROR_UNKNOWN_ISSUER` error

  If you try to open an environment in your browser and Reward's certificate is not working, most probably you have to add Reward's root CA certificate to your browser's CA trust. You can find more information about how to do this in the documentation.

- `Error: ErrCannotFindContainer: container cannot be found: traefik` on Linux using Docker Desktop

  If you use Docker Desktop on Linux you have to configure Reward to use Docker Desktop's socket. See the *Docker Desktop on Linux* documentation page.

- `Error: error terminating mutagen sync: cannot terminate previous sync session: error running command: sh: exit status 1`

  If you see this error message, you have to restart mutagen daemon:

```
mutagen daemon stop
mutagen daemon start
```

- `network reward was found but has incorrect label com.docker.compose.network set to "reward"`

  If you see this error message, you should completely restart reward and remove any leftover networks. The issue occurs because of the different label handling between docker-compose v1 and v2.

```
reward env down
reward svc down
docker network prune

# if the above command doesn't work, you can remove the network manually
# check for existing networks:
docker network ls --filter 'label=dev.reward.network.name'

# remove the network. eg: docker network rm reward
docker network rm __network_name__

reward svc up
reward env up
```

1.9 Experimental Features

1.9.1 WSL2 for Windows

Using WSL2 for Windows with Reward

1.10 Changelog

0.1-beta: Initial Release

Under the hood `docker-compose` is used to control everything which Reward runs (shared services as well as per-project containers) via the Docker Engine.

CHAPTER

TWO

JOIN THE COMMUNITY

Do you have a question? Feel free to ask it! Join Reward's Slack community today!

Join Slack

CHAPTER

THREE

AUTHOR INFORMATION

This project was started in 2021 by Janos Miko.

Like my work?

Buy me a coffee!

CHAPTER

FOUR

ACKNOWLEDGEMENT

Reward was inspired by Warden and most of its functionalities are reused/rewritten in Reward, so HUGE thanks to David Alger and all the contributors for creating it.

- Warden's homepage