wharfrat Documentation

Release REL

Julian Phillips

Sep 30, 2018

Contents:

CHAPTER 1

Introduction

wharfrat is intended to make it easy and convenient to use a development environment in a docker container. The benefits of this are:

- **Simple:** A new development environment is setup with a single simple command, meaning new team members are ready to go immediately.
- **Shared:** Everyone with access to the project (and docker) has access to the development environment.
- **Controlled:** Everyone gets a development environment created from the same image - no more “works for me” issues.
- **Versioned:** The configuration is version controlled, meaning you get the development environment that matches the code branch you are working on.

CHAPTER 2

Installation

…

CHAPTER 3

Basic Usage

…

CHAPTER 4

Configuration

Table of Contents

4.1 Project Configuration

4.2 Crate Configuration

The table below lists the settings available for each crate, their types and default values (if the default is not empty):

cap-add	array of strings	capabilities to enable for the container
cap-drop	array of strings	capabilities to disable for the container
copy-groups	array of strings	groups to copy from the host to the container
env	table of strings	mapping from environment variable name to value
env-blacklist	array of strings	host environment variables to drop
env-whitelist	array of strings	host environment variables to keep
groups	array of strings	groups the user should be in
hostname	string	hostname for container (default: “dev”)
image	string	name of image to create container from
mount-home	bool	should /home be mounted into container (default: true)
ports	array of strings	ports to be exposed from container (-p option to docker)
project-mount	string	path to mount project in container
setup-post	string	script to run in container after unpacking tarballs
setup-pre	string	script to run in container before running tarballs
setup-prep	string	script to run locally before the other setup
shell	string	shell to use in the container
tarballs	table of strings	mapping from tarball location to install location
tmpfs	array of strings	paths in the container where tmpfs should be mounted
volumes	array of strings	list of volume mounts (-v option to docker)
working-dir	string	method to use to set working dir (default: “match”)

cap-add Add additional Linux capabilities to the container. The list of possible values can be found in the docker run reference (https://docs.docker.com/engine/reference/run/#runtime-privilege-and-linux-capabilities).

For example to add the ability to use ptrace inside the container:

```
cap-add = ["SYS_PTRACE"]
```

cap-drop Drop normally enabled Linux capabilities from the container. The list of possible values can be found in the docker run reference (https://docs.docker.com/engine/reference/run/#runtime-privilege-and-linux-capabilities).

For example to drop the ability to bind to privileged ports:

```
cap-drop = ["NET_BIND_SERVICE."]
```

copy-groups TODO . . .

env Specify environment variables to be set in the container. This consists of a table, where the keys are the variable names and the values are the variable values. For example to set SOME_VARIABLE to “some value”:

```
[crates.demo.env]
    "SOME_VARIABLE" = "some value"
```

4.3 Local Configuration

In addition to the shared project configuration each user can have a local configuration. This configuration allows changing the Docker URL, and adding extra steps to the container setup.

```
docker-url = "file:///var/run/docker.sock"

```

(continues on next page)

(continued from previous page)

```
[[setups]]
    project = ".*/test"
    setup-prep = """
        echo "LOCAL PREP: $*"
        pwd
    """

    setup-pre = """
        echo "LOCAL PRE"
        pwd
    """

    setup-post = """
        echo "LOCAL POST"
    """

    [setups.tarballs]
        "path/to/tarball.tgz" = "/path/in/container/to/unpack"

    [setups.env]
        "LOCAL_CRATE_ENV" = "true"

[[setups]]
    setup-prep = """
        echo "LOCAL PREP: $*"
        pwd
    """

    setup-pre = """
        echo "LOCAL PRE"
        pwd
    """

    setup-post = """
        echo "LOCAL POST"
    """

    [setups.env]
        "LOCAL_CRATE_ENV" = "true"
```

The available settings are:

docker-url	The URL to use to connect to Docker	
setups	project	a regular expression that much match the project path for this setup to be applies. If not specified, then ".*" is used.
	crate	a regular expression that must match the crate name for this setup to be applied. If not specified, then ".*" is used.
	setup-prep	script to run locally before doing anything else
	setup-pre	script to run remotely before unpacking tarballs
	setup-post	script to run remotely after unpacking tarballs
	tarballs	a table to tarballs to be unpacked into the container, mapping tarball path to target path in the container
	env	a table of environment variables to set in the container, mapping name to value

CHAPTER 5

Accessing Containers

Table of Contents

5.1 wharfrat run

5.2 Exposing Commands

It is possible to use the `wr-exec` command to expose commands from inside a container to the host. This is normally done by creating an executable config file with a `#!` line that invokes `wr-exec`. For example, if `./test` contains:

```
#!/usr/bin/env wr-exec

project = "/path/to/project/file"
command = ["command", "arg1"]
```

Then, running `./test arg2` will run the command `command arg1 arg2` in the container for the default crate defined in the wharfrat project file at `/path/to/project/file`.

CHAPTER 6

Examples

…

CHAPTER 7

Indices and tables

- genindex
- modindex
- search