Wilfred

Release v0.10.1

Vilhelm Prytz

May 08, 2022

CONTENTS

Wilfred is a command-line interface for running and managing game servers locally. It uses Docker to run game servers in containers, which means they are completely separated. Wilfred can run any game that can run in Docker.

Warning: The documentation has just been migrated to Sphinx. It's currently being updated and things may change at any time. We're also preparing for the Wilfred API to be documented here.

CHAPTER

ONE

INSTALLATION

1.1 Prerequisites

Wilfred currently supports Linux (should be everywhere where Python and Docker is supported), MacOS and now Windows.

Before installing, make sure you have Docker already installed and configured. You can install it from the links below.

- Docker on Linux
- Docker Desktop on MacOS
- Docker Desktop on Windows

You can verify that Docker is installed using `docker --version` or `docker info` (`info` shows more information).

```
user@host:~$ docker --version
Docker version XX.XX.X, build XXXXXXXXXX
```

If you're having trouble accessing the Docker CLI as a non-root user, you can add yourself to the Docker group.

1.2 Homebrew

The recommended way of installing Wilfred is via Homebrew (works on macOS and Linux). Make sure you have it installed on your system. Once Homebrew is installed, use the two commands below to install Wilfred via the offical tap.

```
brew tap wilfred-dev/wilfred
brew install wilfred
```

Want the bleeding edge? You can install the latest commit using `--HEAD` (bugs are to be expected, don't use in production environments!).

```
brew tap wilfred-dev/wilfred
brew install --HEAD wilfred
```

1.3 Pip

Wilfred can be installed using `pip`. You need to use **Python 3.7** or newer to run Wilfred (if you also have `pip2` on your system, run with `pip3`).

```
pip install wilfred --upgrade
```

You can install using a specific python version, e.g. *3.8*.

```
python3.8 -m pip install wilfred --upgrade
```

CHAPTER

TWO

BASIC CONFIGURATION

Once you got Wilfred installed, you can run the setup command to create the basic configuration.

```
wilfred setup
```

Currently, the only config option required is the path for soring data.

```
Path for storing server data [/home/{{ username }}/wilfred-data/servers]:
```

By default, this is `/home/{{ username }}/wilfred-data/servers`. You can use any path as long as you have permissions to access it as the current user.

To create a new server, you can run `wilfred create` and follow the instructions.

CHAPTER

THREE

UPGRADING

To check if you're running the latest version, run `wilfred --version`. If a new version is available, Wilfred will print a message.

If you installed Wilfred using `pip`, then you can upgrade by running the same command as for installing (note the `--upgrade` flag).

```
pip install wilfred --upgrade
```

If you installed Wilfred using `brew`, you can use Homebrew to upgrade Wilfred as you would do with any formula.

```
brew update
brew upgrade
```

3.1 Commands and Syntax

Commands and syntax available for Wilfred. Documentation is automatically generated from source code using sphinx-click

3.1.1 wilfred

Wilfred - A CLI for managing game servers using Docker.

Website - https://wilfredproject.org

Official documentation - https://docs.wilfredproject.org

Source code - https://github.com/wilfred-dev/wilfred

Discord server for support - https://wilfredproject.org/discord

```
wilfred [OPTIONS] COMMAND [ARGS]...
```

Options

--version
Print version and exit

--path
Print paths for configurations and server data

3.2 Images

Not to be confused with actual Docker images, Wilfred images are configuration files used by Wilfred to create game servers. It tells Wilfred which Docker container to run the server in, with which command the server is started with and how to initially install libraries etc.

Wilfred images are formatted in JSON.

This is the configuration file for Vanilla Minecraft.

```
{
    "meta": {
        "api_version": 2
    },
    "uid": "minecraft-vanilla",
    "name": "Vanilla Minecraft",
    "author": "vilhelm@prytznet.se",
    "docker_image": "wilfreddev/java:latest",
    "command": "java -Xms128M -Xmx{{SERVER_MEMORY}}M -jar server.jar",
    "user": "container",
    "stop_command": "stop",
    "default_image": true,
    "config": {
        "files": [
            {
                "filename": "server.properties",
                "parser": "properties",
                "environment": [
                    {
                        "config_variable": "server-port",
                        "environment_variable": "SERVER_PORT",
                        "value_format": null
                    }
                ],
                "action": {
                    "difficulty": "difficulty {}",
                    "white-list": "whitelist {}"
                }
            }
        ]
    },
    "installation": {
        "docker_image": "wilfreddev/alpine:latest",
        "shell": "/bin/ash",
        "script": [
```

(continues on next page)

(continued from previous page)

```
            "apk add curl --no-cache --update jq",
            "if [ \"$MINECRAFT_VERSION\" == \"latest\" ]; then",
            "   VERSION=`curl https://launchermeta.mojang.com/mc/game/version_manifest.
↪json | jq -r '.latest.release'`",
            "else",
                "VERSION=\"$MINECRAFT_VERSION\"",
            "fi",
            "MANIFEST_URL=$(curl -sSL https://launchermeta.mojang.com/mc/game/version_
↪manifest.json | jq --arg VERSION $VERSION -r '.versions | .[] | select(.id== $VERSION␣
↪)|.url')",
            "DOWNLOAD_URL=$(curl ${MANIFEST_URL} | jq .downloads.server | jq -r '. | .url
↪')",
            "curl -o server.jar $DOWNLOAD_URL",
            "if [ \"$EULA_ACCEPTANCE\" == \"true\" ]; then",
            "   echo \"eula=true\" > eula.txt",
            "fi",
            "curl -o server.properties https://raw.githubusercontent.com/wilfred-dev/
↪images/master/configs/minecraft/standard/server.properties",
            "sed -i \"s/{{SERVER_PORT}}/$SERVER_PORT/g\" server.properties",
            "chown -R container:container /server"
        ]
    },
    "variables": [
        {
            "prompt": "Which Minecraft version to use during install?",
            "variable": "MINECRAFT_VERSION",
            "install_only": true,
            "default": "latest",
            "hidden": false
        },
        {
            "prompt": "Do you agree to the Minecraft EULA (https://account.mojang.com/
↪documents/minecraft_eula)?",
            "variable": "EULA_ACCEPTANCE",
            "install_only": true,
            "default": "true",
            "hidden": false
        }
    ]
}
```

3.2.1 Image syntax

All variables are required for image configurations.

- ***meta***
  - *api_version* - Version of configuration.
- *uid* - A unique ID for this config, do not uses spaces. **Must be lowercase.**
- *name* - Name of image to be displayed to user.
- *author* - Email of author.

- *docker_image* - Docker image to run server in.
- *command* - Command to be executed on start.
- *user* - User to run command as, leave empty for default *root*.
- *stop_command* - Command to send to STDIN in order to stop the container.
- *default_image* - Indicates to Wilfred that the image is an official image from the Wilfred project.
- ***config* - Configuration files and how Wilfred should parse them, used within the *wilfred config* command (such as *server.pr***
  - ***files* - List of files to parse.**
    - *filename* - The filename to read and write to.
    - *parser* - What parser Wilfred should use (file-type). Currently, only *properties*, *yaml* and *json* are supported parsers.
    - ***environment* - List of environment variables to link to specific config settings.**
      - *config_variable* - The setting (variable name) as it's named within the configuration file (e.g. *server-port* as that's the name of the setting in *server.properties*).
      - *environment_variable* - A valid environment variable to link with the specified setting. Apart from the variables specified in the image config, *SERVER_PORT* and *SERVER_MEMORY* are valid values.
      - *value_format* - Can be used to append a prefix to the value. Specifying *null* just replaces the value of the setting with the value of the environment variable, without prefixes and suffixes. Otherwise, use *{}* to indicate where the actual value should be set (e.g. *0.0.0.0:{}* is valid syntax).
    - ***action* - Dictionary, sends a command to the STDIN of the container when the setting updates.**
      - *{SETTING_NAME}* - The value should contain the command that should be sent to the container when the specified setting changes. Use *{}* to indicate where the actual value should be set (e.g. *whitelist {}* would send *whitelist on* if the user runs *wilfred config myserver white-list "on"*).
- ***installation***
  - *docker_image* - Docker image to use during installation.
  - *shell* - Shell to use (usually */bin/ash* for Alpine or */bin/bash* for Ubuntu/Debian).
  - *script* - List (array) of commands to execute during installation.
- ***variables* - List of environment variables.**
  - *prompt* - Prompt during server creation/modification.
  - *variable* - Name of environment variable.
  - *install_only* - boolean, variable will only be accessible during installation if *true*.
  - *default* - Default value for prompt, use boolean *true* in order to make variable required but not set a default value and use *""* to make it optional, without default value.
  - *hidden* - Boolean, decides whether this value should be hidden from the user (i.e. static variables).

3.2.2 Environment Variables

Environment variables can be defined in the image configuration. The user will be prompted to enter values for these variables when creating a new server.

The variables are accessible from the installation script and the startup command. But referring to them is slightly different.

To access an environment variable named *MINECRAFT_VERSION* from the installation script, one can use *$MINECRAFT_VERSION* (just as you'd expect it to work).

And to access an environment variable from the startup command, refer to it as *{{image.env.KEY}}* (e.g. *{{image.env.MINECRAFT_VERSION}}* in this case).

3.2.3 Default Variables

The variable *SERVER_MEMORY* and *SERVER_PORT* (so *{{SERVER_MEMORY}}* from the startup command and *$SERVER_MEMORY* from the installation script) are always defined and can be accessed in both the installation script and the startup command.

3.2.4 Default Images

You can find the default images here.

3.2.5 Creating a custom image

When creating a custom image, make sure to **not** put it in the same folder as the default ones. The *default* folder is deleted when Wilfred updates the images from GitHub.

3.3 Wilfred API

> **Warning:** The API is currently not stable and should not be used.

3.3.1 wilfred.api.servers

class wilfred.api.servers.**Servers**

__init__(*docker_client: docker.client.DockerClient*, *configuration: dict*, *images:* wilfred.api.images.Images)

Initiates wilfred.api.Servers, method for controlling servers

Parameters

- **docker_client** (*docker.DockerClient*) – DockerClient object from Docker module
- **configuration** (*dict*) – Dictionary of Wilfred config
- **images** (Images) – wilfred.api.Images object

all(*cpu_load=False*, *memory_usage=False*)

Returns data of all servers

Parameters

- **cpu_load** (*bool*) – Include the CPU load of the container. Defaults to *None* if server is not running.
- **memory_usage** (*bool*) – Include memory usage of the container. Defaults to *None* if server is not running.

remove(*server: wilfred.database.Server*)

Removes specified server

Parameters **server** (*wilfred.database.Server*) – Server database object

console(*server: wilfred.database.Server*, *disable_user_input=False*)

Enters server console

Parameters

- **server** (*wilfred.database.Server*) – Server database object
- **disable_user_input** (*bool*) – Blocks user input if *True*. By default this is *False*.

Raises **ServerNotRunning** – If server is not running

install(*server: wilfred.database.Server*, *skip_wait=False*, *spinner=None*)

Performs installation

Parameters

- **server** (*wilfred.database.Server*) – Server database object
- **skip_wait** (*bool*) – Doesn't stall while waiting for server installation to complete if *True*.
- **spinner** (*Halo*) – If *Halo* spinner object is defined, will then write and perform actions to it.

Raises **WriteError** – If not able to create directory or write to it

kill(*server*)

Kills server container

Parameters **server** (*wilfred.database.Server*) – Server database object

Raises **ServerNotRunning** – If server is not running

command(*server*, *command*)

Sends command to server console

Parameters

- **server** (*wilfred.database.Server*) – Server database object
- **command** (*str*) – The command to send to the stdin of the server

Raises **ServerNotRunning** – If server is not running

sync()

Performs sync, checks for state of containers

rename(*server*, *name*)

Renames server and moves server folder

Parameters

- **server** (*wilfred.database.Server*) – Server database object
- **name** (*str*) – New name of the server

Raises

- **WilfredException** – If server is running
- **WriteError** – If not able to move folder

3.3.2 wilfred.api.images

class wilfred.api.images.**Images**

download(*branch='master'*, *repo='wilfred-dev/images'*)

Downloads default Wilfred Images from GitHub

data_strip_non_ui()

Returns a list of all images with only the variables important to the user shown

Returns Returns `list` of images.

Raises **wilfred.api.images.ImagesNotRead** –

get_image(*uid: str*)

Retrieves image configuration for specific image

Returns Returns `dict` of image configuration.

Raises **wilfred.api.images.ImagesNotRead** –

read_images()

Reads and parses all images on system

Returns Returns `True` if success.

Raises

- **wilfred.api.images.ImagesNotPresent** –
- **wilfred.api.errors.ReadError** –
- **wilfred.api.images.ImageAPIMismatch** –

3.4 Development & Ops

Note: This is mostly used as a reference for the core project team.

3.4.1 Working with Wilfred locally

Make sure you have pipenv installed and that you have cloned the repository to your computer (perhaps created a fork).

At the root of the project, enter the *shell* for the development environment.

```
pipenv shell
```

This will create the environment. Then you need to install the dependencies.

```
pipenv install
pipenv install --dev
```

Instead of using *wilfred <command>* to run Wilfred, you can use the built-in run script.

```
./run.py <command>
```

In this way, you can develop and see your changes instantly.

3.4.2 Publishing a release

Update the version in *wilfred/version.py* and *docs/source/conf.py* accordingly.

Set the version name and release status of this release in *CHANGELOG.md* to *released on YYYY-MM-DD*.

Commit the changes.

Tag the release on GitHub. Include all the changes from *CHANGELOG.md* in the release notes (you can use a previous release as reference).

GitHub Actions should automatically build and release the PyPI package.

Revert the changes in *wilfred/version.py* and commit the changes (should therefore not be included in the release).

3.4.3 Windows

You have to install *pypiwin32* to develop on Windows.

```
pipenv shell
pip install pypiwin32
```

This is not ideal, but due to a bug in Pipenv we cannot put the *pypiwin32* package as a platform specific dependency in the *Pipfile*.

3.4.4 Versioning convention

Wilfred releases should use the semantic versioning convention (i.e. *MAJOR.MINOR.PATCH*).

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

W

INDEX

Symbols

A

C

D

G

I

K

M

R

S

W