Systemd-Service Documentation

Yeison Cardona

Jun 05, 2022

CONTENTS

Simple API to automate the creation of custom daemons for GNU/Linux.

A daemon is a service process that runs in the background and supervises the system or provides functionality to other processes. Traditionally, daemons are implemented following a scheme originating in SysV Unix. Modern daemons should follow a simpler yet more powerful scheme, as implemented by systemd.

Systemd-Service is a Python module to automate the creation of daemons under GNU/Linux environments.

CHAPTER

ONE

INSTALL

```
pip install -U systemd-service
```

CHAPTER

TWO

HANDLE DAEMONS

```
from systemd_service import Service

daemon = Service("stream_rpyc")

daemon.stop()     # Start (activate) the unit.
daemon.start()    # Stop (deactivate) the unit.
daemon.reload()   # Reload the unit.
daemon.restart()  # Start or restart the unit.

daemon.enable()   # Enable the unit.
daemon.disable()  # Disable the unit.

daemon.remove()   # Remove the file unit.
```

This commands are uquivalent to the `systemctl` calls, for example run in terminal the folowing command:

$ systemctl enable stream_rpyc

Can be running inside a Python environment with using `systemd_service`

```
from systemd_service import Service

daemon = Service("stream_rpyc")
daemon.enable()
```

CHAPTER

THREE

CREATING SERVICES

Similar to the previous scripts, the services can be created using `systemd_service`:

```
daemon = Service("stream_rpyc")
daemon.create_service()
```

If the service must be initialized after other service

```
daemon = Service("stream_rpyc")
daemon.create_service(after='ntpd')
```

CHAPTER

FOUR

CREATING TIMERS

Defines a timer relative to when the machine was booted up:

```
daemon = Service("stream_rpyc")
daemon.create_timer(on_boot_sec=15)
```

CHAPTER

FIVE

EXAMPLE

This module is useful when is combined with package scripts declaration in `setup.py` file:

```
# setup.py

scripts=[
    "cmd/stream_rpyc",
]
```

The script could looks like:

```
#!/usr/bin/env python

import sys

if sys.argv[-1] == "systemd":
    from systemd_service import Service
    daemon = Service("stream_rpyc")
    daemon.create_timer(on_boot_sec=10, after='network.target kafka.service')

else:
    from my_module.submodule import my_service
    print("Run 'stream_rpyc systemd' as superuser to create the daemon.")
    my_service()
```

Then the command can be called as a simple script but with the `systemd` argument the command will turn into a service.

```
$ stream_rpyc
# Command executed normally
```

```
$ stream_rpyc systemd
# Service created
```

CHAPTER

SIX

NAVIGATION

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search