Carnival

февр. 29, 2020

Содержимое:

Глава 1

Оборудование (Host)

host.LOCAL_ADDRS = ['local', 'localhost']

class carnival.Host(*addr: str, ssh_user: str = None, ssh_password: str = None, ssh_port=22, ssh_connect_timeout: int = 10, **context*)

Объект, представляющий единицу оборудования.

Carnival не предоставляет никаких сложных абстракций для работы с группами хостов, подразумевая что вы будете использовать встроенные коллекции python и организуете работу так, как будет удобно для вашей задачи.

```
>>> class SetupFrontend(Task):
>>>     def run(self, **kwargs):
>>>         self.step(Frontend(), Host("1.2.3.4", packages=["htop", ]))
```

В более сложных, создать списки в файле *inventory.py*

```
>>> # inventory.py
>>> frontends = [
>>>     Host("1.2.3.4"),
>>>     Host("1.2.3.5"),
>>> ]
```

```
>>> # carnival_tasks.py
>>> import inventory as i
>>> class SetupFrontend(Task):
>>>     def run(self, **kwargs):
>>>         self.step(Frontend(), i.frontends)
```

__init__(*addr: str, ssh_user: str = None, ssh_password: str = None, ssh_port=22, ssh_connect_timeout: int = 10, **context*)

В простом случае, можно передавать хосты прямо в коде файла *carnival_tasks.py*.

Параметры

- **addr** – Адрес сервера

- `ssh_user` – Пользователь SSH
- `ssh_password` – Пароль SSH
- `ssh_port` – SSH порт
- `ssh_connect_timeout` – SSH таймаут соединения
- `context` – Контекст хоста

`host`
Remove user and port parts, return just address

`is_connection_local()` → bool
Check if host's connection is local

Глава 2

Шаги (Host)

class carnival.Step(***context*)

Объект, предназначенный для выполнения группы комманд с какой-то целью. Вызывается из класса *carnival.Task* для выполнения комманд (*carnival.cmd*) на определенных хостах.

Может требовать наличие определенных контекстных переменных для работы, указав их в аргументах метода *run*. Может вернуть значение для дальнейшего использования.

В следующем примере переменная *disk_ name* будет передана в run, а *install* пропущена.

```
>>> host = Host(
>>>     #  Адрес
>>>     "1.2.3.4",
>>>
>>>     # Контекст хоста
>>>     disk_name="/dev/sda1", install=['nginx', 'htop', ]
>>> )
>>> ...
>>> class DiskUsage(Step):
>>>     def run(self, disk_name: str):
>>>         ...
```

__init__(***context*)

Параметры context – Переменные контекста, назначенные при вызове Шага

run(***kwargs*)

Метод который нужно определить для выполнения комманд

Параметры kwargs – Автоматические подставляемые переменные контекста

2.1 Работа с контекстом

Существует несколько видов контекста:

- Специальные переменные контекста

- Контекст хоста (*Host.__init__(..., **context)*)
- Контекст шага (*Step.__init__(**context)*)

Специальные переменные контекста:

- *secrets* - секреты из модуля *secrets*
- *host* - host, на котором вызвано выполнение.

2.2 Перегрузка контекста

Список необходимых переменных контекста для шага строится автоматически из сигнатуры метода *Step.run* из переменных контекста хоста и шага. Специальные переменные контекста имеют самый низкий приоритет. Если существует переменная контекста хоста с таким же именем - она будет иметь более высокий приоритет, и перезапишет значение в аргументе *Step.run*.

Переменные контекста шага имеют самый высокий приоритет, они перезписывают любые другие переменные.

```
>>> host = Host("1.2.3.4", disk_name="/dev/sda1", install=['nginx', 'htop', ])
>>>
>>> class DiskUsage(Step):
>>>     def run(self, disk_name: str):
>>>         return cmd.cli.run(f"du -h {disk_name}")
>>>
>>> class FrontendDiskUsage(Task):
>>>     def run(self):
>>>         # получить использование диска `/dev/sda1` (`host.disk_name`)
>>>         self.step(DiskUsage(), host)
>>>
>>>         # получить использование диска `/dev/sda2`
>>>         # (перегружается из контекста шага)
>>>         self.step(DiskUsage(disk_name="/dev/sda2"), host)
>>>
```

Глава 3

Задача (Task)

class carnival.Task(*dry_run: bool*)

Задача это единица выполнения одного или нескольких шагов на определенных хостах.

Именование задач.

Полное имя задачи состоит из двух частей. <module_name>.<name>. carnival автоматически генерирует имена задач из этих частей, но есть возможность управлять этим вручную, используя два атрибута класса Task.

name: название задачи. если не определено имя будет сгенерировано автоматически. module_name: имя модуля. если назначить пустую строку, полное имя будет включать только название задачи.

call_task(*task_class: Type[Task]*)

Запустить другую задачу Возвращает результат работы задачи

run()

Реализация выполнения задачи

step(*steps: Union[carnival.step.Step, List[carnival.step.Step]], hosts: Union[carnival.host.Host, List[carnival.host.Host]]*) → List[carnival.task.TaskResult]

Запустить шаг(и) на хост(ах) Возвращает объект TaskResult для получения результатов работы каждого шага на каждом хосте

3.1 Простые задачи

class carnival.SimpleTask(*dry_run: bool*)

Запустить шаги *self.steps* на хостах *self.hosts*

3.2 Встроенные задачи

carnival имеет встроенные задачи для удобства использования

class carnival.internal_tasks.Help(*dry_run: bool*)
Показать список доступных задач

3.3 Результат выполнения Task.step

class carnival.task.TaskResult(*host: carnival.host.Host*, *step: carnival.step.Step*, *result: Any*)
Возвращается вызовом метода Task.step

Глава 4

Интерфейс коммандной строки (cli)

`carnival.cli.is_completion_script`(*complete_var: str*) → bool

`carnival.cli.main`()

Глава 5

Шаблоны jinja2 (templates)

`carnival.templates.render`(*template_path: str*, ***context*) → str

Глава 6

Секреты (secrets)

class carnival.secrets_manager.FromCli
Ask in cli

get_secret(*var_name: str*)

class carnival.secrets_manager.FromEnv(*default=None*, *required=False*)
Get from environment

get_secret(*var_name: str*)

exception carnival.secrets_manager.SecretGetError

class carnival.secrets_manager.Static(*value: str*)

get_secret(*var_name: str*)

carnival.secrets_manager.secret(*var_name: str*, *secret_get_method: carnival.secrets_manager.base.SecretGetter*)

Глава 7

Базовые комманды (cmd)

7.1 Apt

`carnival.cmd.apt.force_install`(*pkgname*, *version=None*, *update=False*, *hide=False*)
Install apt package

`carnival.cmd.apt.get_installed_version`(*pkgname: str*) → Optional[str]
Get installed package version Returns None if package not installed

`carnival.cmd.apt.get_pkg_versions`(*pkgname: str*) → List[str]

`carnival.cmd.apt.install`(*pkgname*, *version=None*, *update=True*, *hide=False*) → bool
Install apt package if not installed Returns true if installed, false if was already installed

`carnival.cmd.apt.install_multiple`(**pkg_ names*, *update=True*, *hide=False*) → bool

`carnival.cmd.apt.is_pkg_installed`(*pkgname: str*, *version=None*) → bool
Check is package installed? If version not specified - check any version

`carnival.cmd.apt.remove`(**pkg_ names*, *hide=False*)

7.2 Cli

`carnival.cmd.cli.pty`(*command: str*, ***kwargs*)

`carnival.cmd.cli.run`(*command: str*, ***kwargs*)

7.3 Filesystem

`carnival.cmd.fs.ensure_dir_exists`(*path*, *user=None*, *group=None*, *mode=None*) → None

`carnival.cmd.fs.is_dir_exists`(*dir_ path: str*) → bool

carnival.cmd.fs.is_file_contains(*filename*, *text*, *exact=False*, *escape=True*) → bool

carnival.cmd.fs.is_file_exists(*path*) → bool

carnival.cmd.fs.mkdirs(**dirs*)

7.4 System

carnival.cmd.system.get_current_user_id() → int

carnival.cmd.system.get_current_user_name() → str

carnival.cmd.system.is_current_user_root() → bool

carnival.cmd.system.set_password(*username: str*, *password: str*)

carnival.cmd.system.ssh_authorized_keys_add(*ssh_key: str*, *keys_file='.ssh/authorized_keys'*)

carnival.cmd.system.ssh_authorized_keys_ensure(**ssh_keys*) → None

carnival.cmd.system.ssh_authorized_keys_list() → List[str]

carnival.cmd.system.ssh_copy_id(*pubkey_file='~/.ssh/id_rsa.pub'*) → None

7.5 Systemd

carnival.cmd.systemd.daemon_reload()

carnival.cmd.systemd.disable(*service_name: str*, *reload_daemon=False*, *stop_now=True*)

carnival.cmd.systemd.enable(*service_name: str*, *reload_daemon=False*, *start_now=True*)

carnival.cmd.systemd.restart(*service_name: str*)

carnival.cmd.systemd.start(*service_name: str*, *reload_daemon=False*)

carnival.cmd.systemd.stop(*service_name: str*, *reload_daemon=False*)

7.6 Transfer

carnival.cmd.transfer.get(*remote: str*, *local: str*, *preserve_mode: bool = True*) → <sphinx.ext.autodoc.importer._MockObject object at 0x7f6a25a1f2e8>

carnival.cmd.transfer.put(*local: str*, *remote: str*, *preserve_mode: bool = True*) → <sphinx.ext.autodoc.importer._MockObject object at 0x7f6a25a1f2e8>

carnival.cmd.transfer.put_template(*template_path: str*, *remote: str*, ***context*) → <sphinx.ext.autodoc.importer._MockObject object at 0x7f6a25a1f2e8>

carnival.cmd.transfer.rsync(*source*, *target*, *exclude=()*, *delete=False*, *strict_host_keys=True*, *rsync_opts='–progress -pthrvz'*, *ssh_opts=''*)

Глава 8

Indices and tables

- genindex
- modindex
- search

Содержание модулей Python

C

Алфавитный указатель

Символы

C

D

E

F

G

H

I

L

M

P

R

S

T