
Carnival

февр. 16, 2020

| | | |
|----------|--|-----------|
| 1 | Оборудование (Host) | 1 |
| 2 | Шаги (Host) | 3 |
| 2.1 | Работа с контекстом | 3 |
| 2.2 | Перегрузка контекста | 4 |
| 3 | Задача (Task) | 5 |
| 3.1 | Простые задачи | 5 |
| 3.2 | Встроенные задачи | 5 |
| 3.3 | Результат выполнения Task.step | 6 |
| 4 | Интерфейс командной строки (cli) | 7 |
| 5 | Шаблоны jinja2 (templates) | 9 |
| 6 | Секреты (secrets) | 11 |
| 7 | Базовые команды (cmd) | 13 |
| 7.1 | Apt | 13 |
| 7.2 | Cli | 13 |
| 7.3 | Docker | 13 |
| 7.4 | Filesystem | 14 |
| 7.5 | System | 14 |
| 7.6 | Systemd | 14 |
| 7.7 | Transfer | 14 |
| 8 | Indices and tables | 17 |
| | Содержание модулей Python | 19 |
| | Алфавитный указатель | 21 |

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

```
class carnival.Step(**context)
```

Объект, предназначенный для выполнения группы команд с какой-то целью. Вызывается из класса *carnival.Task* для выполнения команд (*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', 'http', ]
>>> )
>>> ...
>>> 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 команд

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 перезаписывают 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`
>>>         # (перезгружается из контекста шага)
>>>         self.step(DiskUsage(disk_name="/dev/sda2"), host)
>>>
```


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
```

Интерфейс командной строки (cli)

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

Шаблоны jinja2 (templates)

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

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)
```


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.ptty(command: str, **kwargs)`

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

7.3 Docker

`carnival.cmd.docker.install_ce_ubuntu(version=None) → bool`

Returns true if installed, false if was already installed

`carnival.cmd.docker.install_compose(version='1.25.1', dest='/usr/local/bin/docker-compose')`

7.4 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)
```

7.5 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
```

7.6 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)
```

7.7 Transfer

```
carnival.cmd.transfer.get(remote: str, local: str, preserve_mode: bool = True)
    → <sphinx.ext.autodoc.importer._MockObject object at 0x7f5a6fe05b38>
carnival.cmd.transfer.put(local: str, remote: str, preserve_mode: bool = True)
    → <sphinx.ext.autodoc.importer._MockObject object at 0x7f5a6fe05b38>
carnival.cmd.transfer.put_template(template_path: str, remote: str, **context) →
    <sphinx.ext.autodoc.importer._MockObject object at 0x7f5a6fe05b38>
```

```
carnival.cmd.transfer.rsync(source, target, exclude=(), delete=False, strict_host_keys=True,  
                             rsync_opts='-progress -pthruz', ssh_opts="")
```

Indices and tables

- `genindex`
- `modindex`
- `search`

C

- `carnival.cli`, 7
- `carnival.cmd.apt`, 13
- `carnival.cmd.cli`, 13
- `carnival.cmd.docker`, 13
- `carnival.cmd.fs`, 14
- `carnival.cmd.system`, 14
- `carnival.cmd.systemd`, 14
- `carnival.cmd.transfer`, 14
- `carnival.internal_tasks`, 5
- `carnival.secrets_manager`, 11
- `carnival.templates`, 9

СИМВОЛЫ

`__init__()` (метод `carnival.Host`), 1
`__init__()` (метод `carnival.Step`), 3

C

`call_task()` (метод `carnival.Task`), 5
`carnival.cli` (модуль), 7
`carnival.cmd.apt` (модуль), 13
`carnival.cmd.cli` (модуль), 13
`carnival.cmd.docker` (модуль), 13
`carnival.cmd.fs` (модуль), 14
`carnival.cmd.system` (модуль), 14
`carnival.cmd.systemd` (модуль), 14
`carnival.cmd.transfer` (модуль), 14
`carnival.internal_tasks` (модуль), 5
`carnival.secrets_manager` (модуль), 11
`carnival.templates` (модуль), 9

D

`daemon_reload()` (в модуле `carnival.cmd.systemd`), 14
`disable()` (в модуле `carnival.cmd.systemd`), 14

E

`enable()` (в модуле `carnival.cmd.systemd`), 14
`ensure_dir_exists()` (в модуле `carnival.cmd.fs`), 14

F

`force_install()` (в модуле `carnival.cmd.apt`), 13
`FromCli` (класс в `carnival.secrets_manager`), 11
`FromEnv` (класс в `carnival.secrets_manager`), 11

G

`get()` (в модуле `carnival.cmd.transfer`), 14
`get_current_user_id()` (в модуле `carnival.cmd.system`), 14
`get_current_user_name()` (в модуле `carnival.cmd.system`), 14

`get_installed_version()` (в модуле `carnival.cmd.apt`), 13
`get_pkg_versions()` (в модуле `carnival.cmd.apt`), 13
`get_secret()` (метод `carnival.secrets_manager.FromCli`), 11
`get_secret()` (метод `carnival.secrets_manager.FromEnv`), 11
`get_secret()` (метод `carnival.secrets_manager.Static`), 11

H

`Help` (класс в `carnival.internal_tasks`), 5
`host` (атрибут `carnival.Host`), 2
`Host` (класс в `carnival`), 1

I

`install()` (в модуле `carnival.cmd.apt`), 13
`install_ce_ubuntu()` (в модуле `carnival.cmd.docker`), 13
`install_compose()` (в модуле `carnival.cmd.docker`), 13
`install_multiple()` (в модуле `carnival.cmd.apt`), 13
`is_completion_script()` (в модуле `carnival.cli`), 7
`is_connection_local()` (метод `carnival.Host`), 2
`is_current_user_root()` (в модуле `carnival.cmd.system`), 14
`is_dir_exists()` (в модуле `carnival.cmd.fs`), 14
`is_file_contains()` (в модуле `carnival.cmd.fs`), 14
`is_file_exists()` (в модуле `carnival.cmd.fs`), 14
`is_pkg_installed()` (в модуле `carnival.cmd.apt`), 13

L

`LOCAL_ADDRS` (атрибут `carnival.host`), 1

M

`main()` (в модуле `carnival.cli`), 7

`makedirs()` (в модуле *carnival.cmd.fs*), 14

P

`pty()` (в модуле *carnival.cmd.cli*), 13

`put()` (в модуле *carnival.cmd.transfer*), 14

`put_template()` (в модуле *carnival.cmd.transfer*),
14

R

`remove()` (в модуле *carnival.cmd.appt*), 13

`render()` (в модуле *carnival.templates*), 9

`restart()` (в модуле *carnival.cmd.systemd*), 14

`rsync()` (в модуле *carnival.cmd.transfer*), 14

`run()` (метод *carnival.Step*), 3

`run()` (метод *carnival.Task*), 5

`run()` (в модуле *carnival.cmd.cli*), 13

S

`secret()` (в модуле *carnival.secrets_manager*), 11

`SecretGetError`, 11

`set_password()` (в модуле *carnival.cmd.system*), 14

`SimpleTask` (класс в *carnival*), 5

`ssh_authorized_keys_add()` (в модуле
carnival.cmd.system), 14

`ssh_authorized_keys_ensure()` (в модуле
carnival.cmd.system), 14

`ssh_authorized_keys_list()` (в модуле
carnival.cmd.system), 14

`ssh_copy_id()` (в модуле *carnival.cmd.system*), 14

`start()` (в модуле *carnival.cmd.systemd*), 14

`Static` (класс в *carnival.secrets_manager*), 11

`Step` (класс в *carnival*), 3

`step()` (метод *carnival.Task*), 5

`stop()` (в модуле *carnival.cmd.systemd*), 14

T

`Task` (класс в *carnival*), 5

`TaskResult` (класс в *carnival.task*), 6