
Carnival

мар. 20, 2020

1	Оборудование (Host)	1
2	Шаги (Step)	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	Базовые команды (cmd)	11
6.1	Apt	11
6.2	Cli	12
6.3	Filesystem	12
6.4	System	13
6.5	Systemd	14
6.6	Transfer	15
7	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_gateway: Optional[Host] = None, 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_gateway:
Optional[Host] = None, 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
- `ssh_gateway` – Gateway
- `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', '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 команд

Параметры *kwargs* – Автоматические подставляемые переменные контекста, поддерживается ***kwargs*

2.1 Работа с контекстом

Существует несколько видов контекста:

- Специальные переменные контекста
- Переменные окружения, начинающиеся с префикса `CARNIVAL_CTX_`. Переменную `CARNIVAL_CTX_WORKDIR` можно получить по имени `WORKDIR`. Поддерживается `.env-файлы`.
- Контекст хоста (`Host.__init__(..., **context)`).
- Контекст шага (`Step.__init__(**context)`).

Специальные переменные контекста:

- `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, поддерживается `**kwargs`.

- 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 окружения имеют приоритет выше, чем специальные переменные.
- 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.

Именованние 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)

Шаблоны jinja2 (templates)

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

Базовые команды (cmd)

Модуль `carnival.cmd` содержит базовые команды для взаимодействия с сервером. Его цель - оставаться простым и помогать в написании шагов (Step).

Для написания сложных сценариев предполагается использовать шаги(Step).

Основные шаги доступны в отдельном репозитории: <https://github.com/carnival-org/carnival-contrib>.

6.1 Apt

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

Установить пакет без проверки установлен ли он

`carnival.cmd.apt.get_installed_version(pkgname: str) → Optional[str]`

Получить установленную версию пакета

Результат Версия пакета если установлен, *None* если пакет не установлен

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

Получить список доступных версий пакета

`carnival.cmd.apt.install(pkgname, version=None, update=True, hide=False) → bool`

Установить пакет если он еще не установлен в системе

Параметры

- `pkgname` – название пакета
- `version` – версия
- `update` – запустить apt-get update перед установкой
- `hide` – скрыть вывод этапов

Результат *True* если пакет был установлен, *False* если пакет уже был установлен ранее

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

Установить несколько пакетов, если они не установлены

Параметры

- `pkg_names` – список пакетов которые нужно установить
- `update` – запустить `apt-get update` перед установкой
- `hide` – скрыть вывод этапов

Результат `True` если хотя бы один пакет был установлен, `False` если все пакеты уже были установлен ранее

`carnival.cmd.apt.is_pkg_installed(pkgname: str, version=None) → bool`

Проверить установлен ли пакет Если версия не указана - проверяется любая

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

Удалить пакет

Параметры

- `pkg_names` – список пакетов которые нужно удалить
- `hide` – скрыть вывод этапов

6.2 Cli

`carnival.cmd.cli.ptty(command: str, **kwargs) → <sphinx.ext.autodoc.importer._MockObject object at 0x7ff6dca16eb8>`

Запустить команду, используя псевдотерминальную сессию

См <https://docs.pyinvoke.org/en/latest/api/runners.html>

`carnival.cmd.cli.run(command: str, **kwargs) → <sphinx.ext.autodoc.importer._MockObject object at 0x7ff6dca16eb8>`

Запустить команду

6.3 Filesystem

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

Проверить что директория существует и параметры соответствуют заданным

<https://fabric-patchwork.readthedocs.io/en/latest/api/files.html#patchwork.files.directory>

Параметры

- `path` – путь до директории
- `user` – владелец
- `group` – группа
- `mode` – права

`carnival.cmd.fs.is_dir_exists(dir_path: str) → bool`

Узнать существует ли директория

Параметры `dir_path` – путь до директории

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

Содержит ли файл текст См <<https://fabric-patchwork.readthedocs.io/en/latest/api/files.html#patchwork.files.contains>>

Параметры

- `filename` – путь до файла
- `text` – текст который нужно искать
- `exact` – точное совпадение
- `escape` – экранировать ли текст

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

Проверить существует ли файл <<https://fabric-patchwork.readthedocs.io/en/latest/api/files.html#patchwork.files.exists>>

Параметры `path` – путь до файла

`carnival.cmd.fs.makedirs(*dirs) → List[<sphinx.ext.autodoc.importer._MockObject object at 0x7ff6dca2f4a8>]`

Создать директории

Параметры `dirs` – пути которые нужно создать

6.4 System

`carnival.cmd.system.get_current_user_id() → int`

Получить id текущего пользователя

`carnival.cmd.system.get_current_user_name() → str`

Получить имя текущего пользователя

`carnival.cmd.system.is_current_user_root() → bool`

Проверить что текущий пользователь - `root`

`carnival.cmd.system.set_password(username: str, password: str) → <sphinx.ext.autodoc.importer._MockObject object at 0x7ff6dca1e4e0>`

Установить пароль пользователю

Параметры

- `username` – Пользователь
- `password` – Новый пароль

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

Добавить ssh ключ в `authorized_keys`

Параметры

- `ssh_key` – ключ
- `keys_file` – путь до файла `authorized_keys`

Результат `True` если ключ был добавлен, `False` если ключ уже был в файле

`carnival.cmd.system.ssh_authorized_keys_ensure(*ssh_keys) → List[bool]`

Добавить несколько ssh-ключей в авторизованные

Параметры `ssh_keys` – ssh-ключи

Результат Список *True* если ключ был добавлен, *False* если ключ уже был в файле

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

Получить список авторизованных ssh-ключей сервера

```
carnival.cmd.system.ssh_copy_id(pubkey_file=~/.ssh/id_rsa.pub') → bool
```

Добавить публичный ssh-ключ текущего пользователя в авторизованные

Параметры `pubkey_file` – путь до файла с публичным ключем

Результат *True* если ключ был добавлен, *False* если ключ уже был в файле

6.5 Systemd

```
carnival.cmd.systemd.daemon_reload() → <sphinx.ext.autodoc.importer._MockObject object at 0x7ff6dca1e668>
```

Перезагрузить systemd

```
carnival.cmd.systemd.disable(service_name: str, reload_daemon=False, stop_now=True)
→ <sphinx.ext.autodoc.importer._MockObject object at 0x7ff6dca1e668>
```

Убрать сервис из автозапуска

Параметры

- `service_name` – имя сервиса
- `reload_daemon` – перезагрузить systemd
- `stop_now` – Остановить сервис

```
carnival.cmd.systemd.enable(service_name: str, reload_daemon=False, start_now=True)
→ <sphinx.ext.autodoc.importer._MockObject object at 0x7ff6dca1e668>
```

Добавить сервис в автозапуск

Параметры

- `service_name` – имя сервиса
- `reload_daemon` – перезагрузить systemd
- `start_now` – запустить сервис после добавления

```
carnival.cmd.systemd.restart(service_name: str) → <sphinx.ext.autodoc.importer._MockObject object at 0x7ff6dca1e668>
```

Перезапустить сервис

Параметры `service_name` – имя сервиса

```
carnival.cmd.systemd.start(service_name: str, reload_daemon=False) →
<sphinx.ext.autodoc.importer._MockObject object at 0x7ff6dca1e668>
```

Запустить сервис

Параметры

- `service_name` – имя сервиса
- `reload_daemon` – перезагрузить systemd

```
carnival.cmd.systemd.stop(service_name: str, reload_daemon=False) →
                        <sphinx.ext.autodoc.importer._MockObject object at
                        0x7ff6dca1e668>
```

Остановить сервис

Параметры

- `service_name` – имя сервиса
- `reload_daemon` – перезагрузить systemd

6.6 Transfer

```
carnival.cmd.transfer.get(remote: str, local: str, preserve_mode: bool = True)
                        → <sphinx.ext.autodoc.importer._MockObject object at
                        0x7ff6dca1ed30>
```

Скачать файл с сервера <<http://docs.fabfile.org/en/2.5/api/transfer.html#fabric.transfer.Transfer.get>>

Параметры

- `remote` – путь до файла на сервере
- `local` – путь куда сохранить файл
- `preserve_mode` – сохранить права

```
carnival.cmd.transfer.put(local: str, remote: str, preserve_mode: bool = True)
                        → <sphinx.ext.autodoc.importer._MockObject object at
                        0x7ff6dca1ed30>
```

Закачать файл на сервер <<http://docs.fabfile.org/en/2.5/api/transfer.html#fabric.transfer.Transfer.put>>

Параметры

- `local` – путь до локального файла
- `remote` – путь куда сохранить на сервере
- `preserve_mode` – сохранить права

```
carnival.cmd.transfer.put_template(template_path: str, remote: str, **context) →
                        <sphinx.ext.autodoc.importer._MockObject object at
                        0x7ff6dca1ed30>
```

Отрендерить файл с помощью Jinja-шаблонов и закачать на сервер См раздел `templates`.

<<http://docs.fabfile.org/en/2.5/api/transfer.html#fabric.transfer.Transfer.put>>

Параметры

- `template_path` – путь до локального файла Jinja
- `remote` – путь куда сохранить на сервере
- `context` – контекст для рендеринга Jinja2

```
carnival.cmd.transfer.rsync(source, target, exclude=(), delete=False, strict_host_keys=True,
                           rsync_opts='-progress -pthrvz', ssh_opts='')
<https://fabric-patchwork.readthedocs.io/en/latest/api/transfers.html#patchwork.transfers.rsycn>
```

Indices and tables

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

C

`carnival.cmd`, 11
`carnival.cmd.apt`, 11
`carnival.cmd.cli`, 12
`carnival.cmd.fs`, 12
`carnival.cmd.system`, 13
`carnival.cmd.systemd`, 14
`carnival.cmd.transfer`, 15
`carnival.internal_tasks`, 5
`carnival.templates`, 9

СИМВОЛЫ

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

C

`call_task()` (метод *carnival.Task*), 5
carnival.cmd (модуль), 11
carnival.cmd.apt (модуль), 11
carnival.cmd.cli (модуль), 12
carnival.cmd.fs (модуль), 12
carnival.cmd.system (модуль), 13
carnival.cmd.systemd (модуль), 14
carnival.cmd.transfer (модуль), 15
carnival.internal_tasks (модуль), 5
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*),
 12

F

`force_install()` (в модуле *carnival.cmd.apt*), 11

G

`get()` (в модуле *carnival.cmd.transfer*), 15
`get_current_user_id()` (в модуле
carnival.cmd.system), 13
`get_current_user_name()` (в модуле
carnival.cmd.system), 13
`get_installed_version()` (в модуле
carnival.cmd.apt), 11
`get_pkg_versions()` (в модуле *carnival.cmd.apt*),
 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*), 11
`install_multiple()` (в модуле *carnival.cmd.apt*),
 11
`is_connection_local()` (метод *carnival.Host*), 2
`is_current_user_root()` (в модуле
carnival.cmd.system), 13
`is_dir_exists()` (в модуле *carnival.cmd.fs*), 12
`is_file_contains()` (в модуле *carnival.cmd.fs*), 12
`is_file_exists()` (в модуле *carnival.cmd.fs*), 13
`is_pkg_installed()` (в модуле *carnival.cmd.apt*),
 12

L

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

M

`makedirs()` (в модуле *carnival.cmd.fs*), 13

P

`pty()` (в модуле *carnival.cmd.cli*), 12
`put()` (в модуле *carnival.cmd.transfer*), 15
`put_template()` (в модуле *carnival.cmd.transfer*),
 15

R

`remove()` (в модуле *carnival.cmd.apt*), 12
`render()` (в модуле *carnival.templates*), 9
`restart()` (в модуле *carnival.cmd.systemd*), 14
`rsync()` (в модуле *carnival.cmd.transfer*), 15
`run()` (метод *carnival.Step*), 3
`run()` (метод *carnival.Task*), 5
`run()` (в модуле *carnival.cmd.cli*), 12

S

`set_password()` (в модуле *carnival.cmd.system*), 13

`SimpleTask` (класс в *carnival*), 5

`ssh_authorized_keys_add()` (в модуле *carnival.cmd.system*), 13

`ssh_authorized_keys_ensure()` (в модуле *carnival.cmd.system*), 13

`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

`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