Carnival

нояб. 15, 2021

Содержимое:

Глава 1

Оборудование (Host)

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

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

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

```
>>> # inventory.py
>>> frontends = [
>>>     LocalHost(),
>>>     SSHHost("1.2.3.5"),
>>> ]
```

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

carnival.host.AnyConnection = typing.Union[<sphinx.ext.autodoc.importer._MockObject object>, <sphinx.ext

Список адресов которые трактуются как локальное соединение .. deprecated:: 1.4

Host is deprecated, use LocalHost or SSHHost explicitly

class carnival.host.Host(*addr: str*, *ssh_ user: Optional[str] = None*, *ssh_ password: Optional[str] = None*, *ssh_ port: int = 22*, *ssh_ gateway: Optional[SSHHost] = None*, *ssh_ connect_ timeout: int = 10*, *missing_ host_ key_ policy: <sphinx.ext.autodoc.importer._ MockObject object at 0x7fd62b8be640> = <sphinx.ext.autodoc.importer._ MockObject object>*, ***context*)

Параметры

- `addr` – Адрес сервера для SSH или «local» для локального соединения
- `ssh_user` – Пользователь SSH
- `ssh_password` – Пароль SSH
- `ssh_port` – SSH порт
- `ssh_connect_timeout` – SSH таймаут соединения
- `ssh_gateway` – Gateway
- `context` – Контекст хоста

Не рекомендуется, начиная с версии 1.4: Host is deprecated, use LocalHost or SSHHost explicitly

`class carnival.host.LocalHost(**context)`

Локальный хост, работает по локальному терминалу

`host`

Remove user and port parts, return just address

`class carnival.host.SSHHost(addr: str, ssh_user: Optional[str] = None, ssh_password: Optional[str] = None, ssh_port: int = 22, ssh_gateway: Optional[SSHHost] = None, ssh_connect_timeout: int = 10, missing_host_key_policy: <sphinx.ext.autodoc.importer._MockObject object at 0x7fd62b8be640> = <sphinx.ext.autodoc.importer._MockObject object>, **context)`

Удаленный хост, работает по SSH

`is_connection_local()` → bool

Check if host's connection is local

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

`class carnival.host.Host(addr: str, ssh_user: Optional[str] = None, ssh_password: Optional[str] = None, ssh_port: int = 22, ssh_gateway: Optional[SSHHost] = None, ssh_connect_timeout: int = 10, missing_host_key_policy: <sphinx.ext.autodoc.importer._MockObject object at 0x7fd62b8be640> = <sphinx.ext.autodoc.importer._MockObject object>, **context)`

Параметры

- `addr` – Адрес сервера для SSH или «local» для локального соединения
- `ssh_user` – Пользователь SSH
- `ssh_password` – Пароль SSH
- `ssh_port` – SSH порт
- `ssh_connect_timeout` – SSH таймаут соединения
- `ssh_gateway` – Gateway
- `context` – Контекст хоста

Не рекомендуется, начиная с версии 1.4: Host is deprecated, use LocalHost or SSHHost explicitly

`__init__(addr: str, ssh_user: Optional[str] = None, ssh_password: Optional[str] = None, ssh_port: int = 22, ssh_gateway: Optional[SSHHost] = None, ssh_connect_timeout: int = 10, missing_host_key_policy: <sphinx.ext.autodoc.importer._MockObject object at 0x7fd62b8be640> = <sphinx.ext.autodoc.importer._MockObject object>, **context)`

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

`class carnival.host.SSHHost(`*addr: str, ssh_user: Optional[str] = None, ssh_password: Optional[str] = None, ssh_port: int = 22, ssh_gateway: Optional[SSHHost] = None, ssh_connect_timeout: int = 10, missing_host_key_policy: <sphinx.ext.autodoc.importer._MockObject object at 0x7fd62b8be640> = <sphinx.ext.autodoc.importer._MockObject object>, **context*`)`

Удаленный хост, работает по SSH

`__init__(`*addr: str, ssh_user: Optional[str] = None, ssh_password: Optional[str] = None, ssh_port: int = 22, ssh_gateway: Optional[SSHHost] = None, ssh_connect_timeout: int = 10, missing_host_key_policy: <sphinx.ext.autodoc.importer._MockObject object at 0x7fd62b8be640> = <sphinx.ext.autodoc.importer._MockObject object>, **context*`)`

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

`is_connection_local() → bool`

Check if host's connection is local

`class carnival.host.LocalHost(`***context*`)`

Локальный хост, работает по локальному терминалу

`__init__(`***context*`) → None`

Initialize self. See help(type(self)) for accurate signature.

`host`

Remove user and port parts, return just address

Глава 2

Шаги (Step)

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*) → None

Метод который нужно определить для выполнения комманд

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 специальны 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`
>>>         # (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]*) → Any

Запустить другую задачу Возвращает результат работы задачи

run() → Any

Реализация выполнения задачи

step(*steps: Union[carnival.step.Step, List[carnival.step.Step]], hosts: Union[carnival.host.LocalHost, carnival.host.SSHHost, List[Union[carnival.host.LocalHost, carnival.host.SSHHost]]]*) → 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

help = 'List available commands and help'

module_name = ''

run() → None
Реализация выполнения задачи

3.3 Результат выполнения Task.step

class carnival.task.TaskResult(*host: Union[carnival.host.LocalHost, carnival.host.SSHHost], step: carnival.step.Step, result: Any*)
Возвращается вызовом метода Task.step

Глава 4

Интерфейс коммандной строки (cli)

carnival.cli.except_hook(*type: Type[Any], value: Any, traceback: Any*) → None

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

carnival.cli.main() → int

```
>>> $ poetry run python -m carnival --help
>>> Usage: python -m carnival [OPTIONS] {help|test}...
>>> Options:
>>> -d, --dry_run  Simulate run
>>> --debug        Turn on debug mode
>>> --help         Show this message and exit.
```

Глава 5

Шаблоны jinja2 (templates)

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

Глава 6

Базовые комманды (cmd)

Модуль carnival.cmd содержит базовые комманды 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: str*, *version: Optional[str] = None*, *update: bool = False*, *hide: bool = False*) → None

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: str*, *version: Optional[str] = None*, *update: bool = True*, *hide: bool = 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: bool = True, hide: bool = 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: Optional[str] = None) → bool
Проверить установлен ли пакет Если версия не указана - проверяется любая

carnival.cmd.apt.remove(*pkg_names, hide: bool = False) → None
Удалить пакет

Параметры

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

6.2 Cli

carnival.cmd.cli.pty(command: str, **kwargs) → <sphinx.ext.autodoc.importer._MockObject object at 0x7fd62b8da670>
Запустить комманду, 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 0x7fd62b8da670>
Запустить комманду

6.3 Filesystem

carnival.cmd.fs.ensure_dir_exists(path: str, user: Optional[str] = None, group: Optional[str] = None, mode: Optional[str] = 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: str*, *text: str*, *exact: bool = False*, *escape: bool = 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: str*) → 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.mkdirs(**dirs*) → List[<sphinx.ext.autodoc.importer._MockObject object at 0x7fd62b867e80>]

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 0x7fd62b8daca0>

Установить пароль пользователю

Параметры

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

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

Добавить ssh ключ в *authorized_keys*

Параметры

- `ssh_key` – ключ
- `keys_file` – пусть 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: str = '~/.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 0x7fd62b8dad90>
Перегрузить systemd

`carnival.cmd.systemd.disable(`*service_name: str*, *reload_daemon: bool = False*, *stop_now: bool = True*`)` → <sphinx.ext.autodoc.importer._MockObject object at 0x7fd62b8dad90>
Убрать сервис из автозапуска

Параметры

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

`carnival.cmd.systemd.enable(`*service_name: str*, *reload_daemon: bool = False*, *start_now: bool = True*`)` → <sphinx.ext.autodoc.importer._MockObject object at 0x7fd62b8dad90>
Добавить сервис в автозапуск

Параметры

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

`carnival.cmd.systemd.restart(`*service_name: str*`)` → <sphinx.ext.autodoc.importer._MockObject object at 0x7fd62b8dad90>
Перезапустить сервис

Параметры `service_name` – имя сервиса

`carnival.cmd.systemd.start(`*service_name: str*, *reload_daemon: bool = False*`)` → <sphinx.ext.autodoc.importer._MockObject object at 0x7fd62b8dad90>
Запустить сервис

Параметры

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

carnival.cmd.systemd.stop(*service_name: str*, *reload_daemon: bool = False*) → <sphinx.ext.autodoc.importer._MockObject object at 0x7fd62b8dad90>

Остановить сервис

Параметры

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

6.6 Transfer

carnival.cmd.transfer.get(*remote: str*, *local: str*, *preserve_mode: bool = True*) → <sphinx.ext.autodoc.importer._MockObject object at 0x7fd62b867940>

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 0x7fd62b867940>

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 0x7fd62b867940>

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` – контекс для рендеринга jinja2

carnival.cmd.transfer.rsync(*source: str*, *target: str*, *exclude: Iterable[str] = ()*, *delete: bool = False*, *strict_host_keys: bool = True*, *rsync_opts: str = '–progress -pthrvz'*, *ssh_opts: str = ''*) → <sphinx.ext.autodoc.importer._MockObject object at 0x7fd62b867940>

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

Глава 7

Indices and tables

- genindex
- modindex
- search

Содержание модулей Python

C

Алфавитный указатель

Символы

A

C

D

E

F

G

H

I

L

M

P

R

S

T