# Модуль decs_vm для управления облачными ресурсами посредством Ansible

Сергей Шубин © Digital Energy Cloud Solutions, 2018

Email: sergey.shubin@digitalenergy.online / svs1370@gmail.com

Актуальные версии модуля и документацию см. на github.com/rudecs/decsamo

## Содержание

Предисловие ..... 2
«Короче, Склифосовский!» ..... 3
Обзор облачной платформы DECS ..... 5
Основные понятия ..... 5
Способы авторизации ..... 6
Модуль decs_vm ..... 7
Назначение ..... 7
Системные требования ..... 7
Установка ..... 7
Параметры ..... 8
Возвращаемые значения ..... 16
Типовые сценарии и примеры использования ..... 17
Как создать новый сервер в новом виртуальном сетевом сегменте. ..... 17
Как создать новый сервер в существующем виртуальном сетевом сегменте. ..... 18
Как получить и использовать информацию о созданном сервере внутри playbook ..... 20
Как управлять правилами трансляции сетевых портов сервера. ..... 21
Как управлять подключением / отключением внешнего IP адреса. ..... 23
Как включать / выключать существующий сервер. ..... 24
Как изменить CPU, RAM и размер загрузочного диска существующего сервера. ..... 25
Как удалить существующий сервер. ..... 26
Как выглядит task для разных режимов авторизации. ..... 27

# Предисловие

В данном документе описывается, как с помощью модуля *decs_vm* и системы Ansible управлять созданием и конфигурированием облачных ресурсов в платформе Digital Energy Cloud Solutions (“DECS”).

Если вы хорошо знакомы с системой Ansible и хотите максимально быстро начать использовать виртуальные серверы на платформе DECS в своих Ansible playbooks, то можете сразу перейти к разделу «Короче, Склифосовский!». Если у вас все же возникнут вопросы по облачной платформе DECS и порядку авторизации в ней, то обратитесь к главе «Обзор облачной платформы DECS», разделы «Основные понятия» и «Способы авторизации».

Если вы только начинаете использовать систему Ansible и облачную платформу DECS, то рекомендуем вам начать с главы «Обзор облачной платформы DECS», после чего изучить главу «Модуль decs_vm».

Чтобы узнать, как инсталлировать модуль *decs_vm* в вашу систему Ansible, обратитесь к разделам «Системные требования» и «Установка».

## «Короче, Склифосовский!»

Данный раздел предназначен для тех, кто хорошо знаком с системой Ansible, а также имеет представление об основных понятиях и способах авторизации в облачной платформе DECS.

Ниже приведен пример Ansible playbook, который создает виртуальный сервер, обновляет in-memory inventory информацией о только что созданном сервере и выполняет простейшую команду на уровне его гостевой ОС.

```
---
- hosts: ansible_master
  tasks:
  - name: obtain VM with port forward to SSH and direct IP address
    decs_vm:
      authenticator: oauth2
      app_id: "{{ my_app_id }}"
      app_secret: "{{ my_app_secret }}"
      oauth2_url: https://itsyou.online
      controller_url: "https://ds1.digitalenergy.online"
      name: MyFirstVM
      state: present
      cpu: 2
      ram: 4096
      boot_disk:
        size: 10
      image_name: "Ubuntu 16.04"
      port_forwards:
        - ext_port: 21022
          int_port: 22
          proto: tcp
      ext_network: present
      tenant: "MyTenant"
      vdc_name: "MyFirstVDC"
      annotation: "My 1st VM created and managed with DECS VM module"
    delegate_to: localhost
    register: new_vm

  - name: add new VM to the in-memory inventory into group just_created
    add_host:
      groups: just_created
      hostname: "{{ new_vm.vm_facts.name }}"
      ansible_host: "{{ new_vm.vm_facts.vdc_ext_ip }}"
      ansible_port: 21022
      ansible_user: "{{ new_vm.vm_facts.username }}"
      ansible_password: "{{ new_vm.vm_facts.password }}"

- hosts: just_created
  tasks:
  - name: check connectivity to the new VM by running a remote command on
it
    command: uname -a

...
```

Виртуальный сервер в облачной платформе DECS создается посредством *task*, которая вызывает модуль[1] *decs_vm* с минимальным набором необходимых параметров, а именно:

1) Параметры, описывающие подключение к облачной платформе. В нашем примере выбран режим авторизации[2] `oauth2`, для которого необходимо указать application ID, application secret и URL, по которому доступен авторизационный сервер, работающий по протоколу Oauth2. Независимо от режима авторизации указывается URL контроллера, который управляет соответствующим экземпляром платформы.
2) Характеристики создаваемого виртуального сервера (имя “MyFirstVM”, 2 CPU, 4ГБ RAM, загрузочный диск объемом 10ГБ, название образа гостевой операционной системы «Ubuntu 16.04»). После успешного создания виртуального сервера его характеристики записываются[3] в переменную `new_vm` и будут доступны в процессе дальнейшего исполнения playbook-а как словарь `new_vm.vm_facts`.
3) Параметры `tenant` и `vdc_name`, описывающие принадлежность создаваемого сервера к подписчику (“MyTenant”) и идентифицированному по имени виртуальному сетевому сегменту (“MyFristVDC”) в облаке[4] DECS, находящимся под управлением контроллера, доступного по адресу `controller_url`.

Второй task в приведенном примере динамически обновляет Ansible in-memory inventory, добавляя в группу хостов `just_created` информацию о только что созданном виртуальном сервере.

Третий task выполняет команду `uname -a` на уровне гостевой ОС виртуального сервера.

Обратите внимание, что модуль *decs_vm* выполняется локально на Ansible-сервере (директива `delegate_to: localhost` в первом task).

[1] Модуль *decs_vm* должен быть инсталлирован на вашем Ansible-сервере. Порядок инсталляции см. в разделе «Установка».
[2] Подробности см. в разделе «Способы авторизации».
[3] Структуру словаря см. в разделе «Возвращаемые значения».
[4] Основные понятия и концепции облачной платформы DECS см. в главе «Обзор облачной платформы DECS».

# Обзор облачной платформы DECS

## Основные понятия

Ниже перечислены основные понятия с указанием соответствующих им параметров в модуле *decs_vm*.

- Контроллер облачной инфраструктуры DECS – управляющее приложение, которое обеспечивает авторизацию пользователей и оркестрацию облачных ресурсов.
  Адрес контроллера задается в обязательном параметре `controller_url`.
- Авторизационный провайдер – приложение, работающее по протоколу Oauth2, предназначенное для выпуска и валидации токенов доступа к контроллеру облачной инфраструктуры.
  Адрес авторизационного провайдера задается в параметре `oauth2_url`.
- Подписчик (*tenant*) – сущность, которая используется для группирования облачных ресурсов по принадлежности к определенному клиенту для целей учета потребления и биллинга. Имя подписчика задается параметром `tenant`.
- Пользователь (*user*) – пользователь облачной инфраструктуры, представленный учетной записью. Чтобы получить возможность управлять облачными ресурсами (например, создавать виртуальные сервера) пользователь должен быть ассоциирован с одним или несколькими подписчиками и иметь соответствующие права, определяемые ролевой моделью, принятой в облачной платформе DECS.
  Для доступа к платформе пользователь должен авторизоваться одним из способов, описанных в разделе «Способы авторизации».
- Виртуальный сетевой сегмент – защищенный сетевой сегмент, служащий для сетевого подключения виртуальных серверов и группирования их по функциональному признаку. Данный сегмент можно рассматривать как небольшой персональный дата-центр (*virtual data center, VDC*), в котором размещаются группы виртуальных серверов.
  Виртуальный сетевой сегмент идентифицируется одним из двух способов:
  - По уникальному в рамках каждого контроллера DECS цифровому идентификатору (параметр `vdc_id`).
  - По комбинации параметров `tenant` и `vdc_name`. Обратите внимание, что имя *`vdc_name`* уникально только в рамках одного и того же `tenant`.
- Виртуальный сервер – виртуальная машина, которая работает в облаке DECS и доступна по сети. Виртуальный сервер характеризуется количеством выделенных ему CPU (параметр `cpu`), объемом ОЗУ (`ram`), размером загрузочного диска (`bootdisk.size`). При создании виртуального сервера на загрузочный диск устанавливается образ операционной системы, заданный в параметре `image_name`.
  Помимо загрузочного диска к виртуальному серверу можно подключить несколько дисков для хранения прикладных данных, конфигурация которых задается параметром `data_disks`.
  Сетевой доступ к виртуальному серверу возможен через настройку правил трансляции сетевых портов на периметре соответствующего виртуального сетевого сегмента (параметр `port_forwarding`), а также посредством назначения серверу прямого внешнего IP адреса (параметры `ext_network` и `ext_network_id`).
  Виртуальный сервер идентифицируется одним из двух способов:
  - По уникальному в рамках каждого контроллера DECS цифровому идентификатору сервера (параметр `id`).

- По комбинации параметров `name`, `tenant` и `vdc_name` (или `vdc_id`). Обратите внимание, что имя виртуального сервера `name` уникально только в рамках одного и того же виртуального сетевого сегмента.

## Способы авторизации

На текущий момент облачная платформа DECS поддерживает два базовых типа авторизации:

1. С использованием авторизационного провайдера, работающего по протоколу Oauth2. Данный способ является предпочтительным, так как обеспечивает бОльшую гибкость и безопасность. Для авторизации в этом режиме в *task* необходимо указать параметры `oauth2_url` и `controller_url`, а также предоставить одно из нижеперечисленного:
    a. Комбинация *Application ID & Application secret*, соответствующих пользователю, от имени которого будет осуществляться управление облачными ресурсами в текущей сессии. В процессе проверки предоставленных модуль получает от авторизационного провайдера токен (JSON Web Token), который затем используется для доступа к указанному DECS контроллеру.
    Чтобы модуль *decs_vm* авторизовался по данному варианту, в *task* следует установить параметр `authenticator: oauth2` и задать параметры `app_id` и `app_secret`.
    b. *JSON Web Token* – заранее полученный от авторизационного провайдера токен доступа, ассоциированный с определенным пользователем, от имени которого будет осуществляться управление облачными ресурсами в текущей сессии.
    Чтобы модуль *decs_vm* авторизовался по данному варианту, в *task* следует установить параметр `authenticator: jwt` и задать параметр `jwt`.
2. С использованием комбинации *user:password*. Данный режим не использует внешних авторизационных провайдеров и подразумевает, что пользователь с такой комбинацией *user:password* зарегистрирован непосредственно на указанном в параметре `controller_url` контроллере облачной инфраструктуры DECS.
Чтобы модуль *decs_vm* авторизовался по данному варианту, в *task* следует установить параметр `authenticator: legacy` и задать параметры `user` и `password`.

После успешной авторизации пользователь получает доступ к ресурсам, находящимся под управлением соответствующего DECS контроллера. Доступ предоставляется в рамках подписчиков, с которыми ассоциирован данный пользователь, и в соответствии с присвоенными ему ролями.

# Модуль decs_vm

## Назначение

Модуль *decs_vm* предназначен для выполнения следующих действий над виртуальными серверами в облачной платформе DECS:

- Создание нового виртуального сервера.
- Изменение конфигурации существующего виртуального сервера:
  - Изменение количества выделенных CPU и виртуальной ОЗУ. Следует иметь ввиду, что изменение этих параметров в *меньшую* сторону может потребовать перезагрузки гостевой ОС.
  - Увеличение размера загрузочного диска.
  - Добавление, редактирование и удаление правил трансляции сетевых портов (управление port forwarding rules).
  - Назначение и удаление прямого внешнего IP адреса.
- Изменение состояния существующего виртуального сервера:
  - Выключение / включение.
  - Перезагрузка, приостановка / возобновление работы гостевой ОС.
- Удаление существующего виртуального сервера.

## Системные требования

Для корректного функционирования модуля *decs_vm* на сервере требуется наличие следующего программного обеспечения:

- Python модуль *decs_utility.py* (содержит библиотечные функции, поддерживающие функциональность Ansible-модулей для управления облачными ресурсами DECS).
- Python модуль *PyJWT* для работы с JSON Web Tokens (JWT).
- Ansible версии 2.2 или выше.
- Интерпретатор языка Python версии 2.6 или выше.

## Установка

Актуальный исходный код модуля *decs_vm* и необходимых для его работы библиотечных модулей на языке Python доступен по адресу https://github.com/rudecs/decsamo .

Чтобы установить модуль *decs_vm* на Ansible сервер[5], выполните следующие шаги:

1) Скачайте из папкие *Modules* по адресу https://github.com/rudecs/decsamo файлы *decs_vm.py* и *decs_utility.py*.
2) Поместите файл *decs_vm.py* в папку /usr/share/ansible/plugins/modules/ на Ansible сервере. Проверьте, что права доступа к файлу допускают его выполнение.
3) Поместите файл *decs_utility.py* в папку /usr/lib/python2.7/dist-packages/ansible/module_utils/ на Ansible сервере. Проверьте, что права доступа к файлу допускают его выполнение.

[5] Пути к целевому расположению файлов на Anisble сервере даны в предположении ОС Ubuntu 16.x и Ansible версии 2.6. Для других версий ОС, Ansible и Python пути могут отличаться от указанных.

## Параметры

Ниже в алфавитном порядке приведен полный список параметров для модуля *decs_vm*. Актуальную информацию по параметрам, которые поддерживает версия модуля, установленного на вашем Ansible-сервере, можно получить командой:

```
ansible-doc -t module decs_vm
```

| **Параметр** | **Тип, допустимые значения и установки по умолчанию** | **Описание** |
|---|---|---|
| annotation | (string) | Опциональное описание виртуального сервера. Этот параметр используется только в момент создания нового виртуального сервера и игнорируется при любых действиях над существующими серверами. |
| app_id | (string) | Идентификатор приложения, использующийся для подключения к контроллеру облачной платформы DECS в режиме `authenticator: oauth2`<br>Данный параметр является обязательным для указанного режима.<br>Если параметр не задан в playbook, модуль *decs_vm* будет использовать значение переменной окружения `DECS_APP_ID`. |
| app_secret | (string) | Секретный ключ приложения, который используется для подключения к контроллеру облачной платформы DECS в режиме `authenticator: oauth2`<br>Данный параметр является обязательным для указанного режима.<br>Так как он содержит секретную информацию, то его не рекомендуется задавать непосредственно в playbook.<br>Если параметр не задан в playbook, то модуль *decs_vm* будет использовать значение переменной окружения `DECS_APP_SECRET`. |
| authenticator | **Значения**:<br>legacy<br>oauth2<br>jwt <- *default* | Режим аутентификации при подключении к контроллеру облачной платформы DECS.<br>Параметр является обязательным. |

| Параметр | Тип, допустимые значения и установки по умолчанию | Описание |
|---|---|---|
| boot_disk | (dict) | Конфигурация загрузочного диска виртуального сервера.<br>Данный параметр является обязательным при создании нового виртуального сервера. Для существующих виртуальных серверов этот параметр является опциональным – с его помощью можно увеличить размер загрузочного диска.<br>Параметр представляет собой словарь со следующими ключами:<br>• (int) `size` – размер диска в GB.<br>• (string) `model` – модель системы хранения данных, на базе которой будет создан данный загрузочный диск. Допустимые модели: ‘ovs’, ‘iscsi’<br>• (string) `pool` – пул дисковых ресурсов, из которого будет предоставлен данный загрузочный диск. Имена пулов могут меняться от одной модели СХД к другой. Если указанный пул не найден, модуль ожидает, что платформа создаст дисковый ресурс в “default” пуле, который всегда должен присутствовать. |
| controller_url | (string) | URL контроллера, соответствующего экземпляру облачной платформы DECS, в рамках которого должен быть создан (или уже существует) данный виртуальный сервер.<br>Данный параметр является обязательным. |
| cpu | (int) | Количество виртуальных CPU в виртуальном сервере.<br>Параметр является обязательным при создании нового сервера, во всех других случаях он опциональный. Если указать его для уже существующего сервера, то будет выполнена попытка изменить количество CPU. Следует иметь ввиду, что *уменьшение* количества CPU у работающего сервера, как правило, потребует его перезагрузки. |

| Параметр | Тип, допустимые значения и установки по умолчанию | Описание |
|---|---|---|
| datacenter | (string) | Целевой *datacenter* под управлением заданного DECS контроллера, где размещается данный виртуальный сервер.<br>Этот параметр является обязательным при создании нового виртуального сервера, когда вместе с сервером также требуется создать виртуальный сетевой сегмент, в котором он будет расположен.<br>При всех прочих операциях данный параметр игнорируется. |
| data_disks | (list of dict) | Список data-дисков, которые надо создать и презентовать виртуальному серверу.<br>Данный параметр является опциональным, используется только при создании нового виртуального сервера и игнорируется при любых других операциях.<br>Параметр представляет собой словарь со следующими ключами:<br>• (int) `size` – размер диска в GB.<br>• (string) `model` – модель системы хранения данных, на базе которой будет создан данный data-диск. Допустимые модели: '*ovs*', '*iscsi*'<br>• (string) `pool` – пул дисковых ресурсов, из которого будет предоставлен данный data-диск. Имена пулов могут меняться от одной модели СХД к другой. Если указанный пул не найден, модуль ожидает, что платформа создаст дисковый ресурс в "*default*" пуле, который всегда должен присутствовать. |
| ext_network | **Значения**:<br>present<br>absent <- *default* | Данный опциональный параметр задает наличие/отсутствие прямого сетевого подключения данного виртуального сервера к внешнему сетевому сегменту (выделение публичного IP адреса). |

| Параметр | Тип, допустимые значения и установки по умолчанию | Описание |
|---|---|---|
| ext_network_id | (int) | Данный опциональный параметр используется в паре с `ext_network`, чтобы задать идентификатор внешней сети, из которой следует брать внешний IP адрес.<br>Если этот параметр опущен при *`ext_network: present`*, то внешний IP адрес будет выбран из внешнего сетевого сегмента по умолчанию. |
| id | (int) | Уникальный цифровой идентификатор виртуального сервера внутри платформы.<br>Этот параметр используется как один из методов идентификации существующего сервера (альтернатива – по комбинации `name`, `vdc_name` и `tenant`) и игнорируется при создании нового сервера, так как для нового сервера платформа назначает этот идентификатор автоматически.<br>Если при вызове модуля *decs_vm* существующий виртуальный сервер идентифицируется по своему `id`, то параметры `tenant`, `vdc_name` и `vdc_id` игнорируются. |
| image_name | (string) | Название образа ОС, на базе которого следует создать виртуальный сервер.<br>Этот параметр является обязательным при создании нового виртуального сервера и игнорируется при любых других операциях.<br>Параметр должен задаваться в точном соответствии с тем, как назван нужный образ ОС в облачной инфраструктуре (с соблюдением заглавных и строчных символов, а также пробелов). |

| Параметр | Тип, допустимые значения и установки по умолчанию | Описание |
|---|---|---|
| jwt | (string) | JSON Web Token (JWT), который будет использоваться для подключения к контроллеру облачной платформы DECS в режиме `authenticator: jwt`<br>Данный параметр является обязательным для указанного режима.<br>Так как он содержит потенциально секретную информацию, а сам JWT, как правило, имеет ограниченное время жизни, то его не рекомендуется задавать непосредственно в playbook.<br>Если этот параметр не определен в playbook, то модуль *decs_vm* будет использовать значение переменной окружения `DECS_JWT`. |
| name | (string) | Название виртуального сервера.<br>Чтобы модуль *decs_vm* мог управлять сервером по его названию, также необходимо задать комбинацию `tenant` and `vdc_name` или идентификатор `vdc_id`.<br>Параметр должен задаваться в точном соответствии с тем, как назван сервер в облачной инфраструктуре (с соблюдением заглавных и строчных символов, а также пробелов).<br>Если для существующего виртуального сервера указаны и `name`, и `id`, то параметр `name` игнорируется и идентификация сервера выполняется по его `id`. |
| oauth2_url | (string) | URL авторизационного сервера, работающего по протоколу oauth2, который должен использоваться в режиме `authenticator: oauth2`<br>Данный параметр является обязательным для указанного режима.<br>Если параметр не задан в playbook, модуль *decs_vm* будет использовать значение переменной окружения `DECS_OAUTH2_URL`. |

| Параметр | Тип, допустимые значения и установки по умолчанию | Описание |
|---|---|---|
| password | (string) | Пароль для подключения к контроллеру облачной инфраструктуры DECS в режиме `authenticator: legacy`<br>Данный параметр является обязательным для указанного режима.<br>Так как он содержит секретную информацию, то его не рекомендуется задавать в playbook.<br>Если параметр не задан в playbook, то модуль *decs_vm* будет использовать значение переменной окружения `DECS_PASSWORD`. |
| port_forwards | (list of dict) | Список правил настройки *port forwarding* для виртуального сервера.<br>Каждое правило в списке это словарь со следующими ключами:<br>• (int) `ext_port` – номер внешнего сетевого порта;<br>• (int) `int_port` – номер внутреннего сетевого порта;<br>• (str) `proto` – наименование протокола. Доступные протоколы: '*tcp*', '*udp*'. |
| ram | (int) | Объем оперативной памяти (RAM) в MB, выделенной данному виртуальному серверу.<br>Параметр является обязательным при создании нового сервера. Если указать его для уже существующего сервера, то будет выполнена попытка изменить объем выделенной серверу памяти.<br>Следует иметь ввиду, что *уменьшение* объема памяти работающего сервера в большинстве случаев потребует его перезагрузки. |
| ssh_key | (string) | Открытая часть SSH ключа, который необходимо авторизовать на создаваемом виртуальном сервере для пользователя, заданного параметром `ssh_key_user`.<br>Данный параметр применим только для Linux серверов, является опциональным, используется только при создании нового сервера и игнорируется при других операциях. |

| Параметр | Тип, допустимые значения и установки по умолчанию | Описание |
|---|---|---|
| ssh_key_user | (string) | Имя пользователя на уровне гостевой ОС (только для Linux серверов) для которого авторизуется SSH ключ, заданный параметром `ssh_key`. Данный параметр является обязательным, если задан `ssh_key`, используется только при создании нового сервера и игнорируется при других операциях. |
| state | **Значения**:<br>present <- *default*<br>absent<br>poweredon<br>poweredoff<br>paused | Целевое состояние виртуального сервера на выходе из модуля *decs_vm*. |
| tags | (string) | Строка, содержащая набор текстовых меток, которые надлежит присвоить данному виртуальному серверу.<br>Данные текстовые метки представляют собой произвольный текст, который можно использовать для группировки и индексирования виртуальных серверов во внешних приложениях. |
| tenant | (string) | Имя подписчика, которому будет принадлежать новый виртуальный сервер (или уже принадлежит существующий).<br>Параметр должен задаваться в точном соответствии с тем, как назван нужный подписчик в облачной инфраструктуре (с соблюдением заглавных и строчных символов, а также пробелов).<br>Этот параметр является опциональным и используется в сценариях, когда вместе с новым виртуальным сервером создается новый виртуальный облачный сегмент, в котором будет размещаться данный сервер, или когда уже существующий целевой сегмент задается комбинацией `tenant` и `vdc_name`. |

| Параметр | Тип, допустимые значения и установки по умолчанию | Описание |
|---|---|---|
| user | (string) | Имя пользователя, непосредственно зарегистрированного на контроллере облачной инфраструктуры DECS, которое используется для подключения к контроллеру в режиме `authenticator: legacy`<br>Данный параметр является обязательным для указанного режима.<br>Если параметр не задан в playbook, модуль *decs_vm* будет использовать значение переменной окружения `DECS_USER`. |
| vdc_id | (int) | Уникальный цифровой идентификатор уже существующего виртуального сетевого сегмента (VDC), где будет создан новый или находится уже существующий виртуальный сервер.<br>Данный параметр является одним из методов идентификации существующего VDC (альтернативой является задание комбинации `tenant` и `vdc_name`). |
| vdc_name | (string) | Имя виртуального сетевого сегмента (VDC), где будет создан новый или находится уже существующий виртуальный сервер.<br>Данный параметр является одним из методов идентификации существующего VDC, когда задается пара `tenant` и `vdc_name` (альтернативой является задание `vdc_id`, однако такой метод применим только для уже существующих VDC).<br>Параметр должен задаваться в точном соответствии с тем, как назван нужный сегмент в облачной инфраструктуре (с соблюдением заглавных и строчных символов, а также пробелов).<br>Если заданы и `vdc_id`, и `vdc_name`, то параметр `vdc_name` игнорируется. |
| workflow_callback | (string) | URL, по которому вышестоящее приложение (например, пользовательский портал или оркестратор верхнего уровня, инициирующий запуск Ansible playbook) ожидает API вызова, в параметрах которого модуль *desc_vm* будет оперативно передавать информацию о своем статусе и текущей фазе исполнения.<br>Данный параметр является опциональным. |

| Параметр | Тип, допустимые значения и установки по умолчанию | Описание |
|---|---|---|
| workflow_context | (string) | Контекстная информация, которая будет содержаться в параметрах API вызова, адресованного к `workflow_callback` URL. Данная информация призвана однозначно идентифицировать задачу, выполняемую модулем в настоящий момент, чтобы оркестратор верхнего уровня мог сопоставить получаемые через вызов `workflow_callback` данные со своим внутренним состоянием и отслеживать инициированные им задачи. Параметр является опциональным и имеет значение только при условии, что также задан `workflow_callback`. |

## Возвращаемые значения

Модуль возвращает информацию о виртуальном сервере в виде словаря *`vm_facts`* со следующими ключами:

| Ключ | Тип данных | Описание |
|---|---|---|
| ext_gateway | string | IP адрес шлюза по умолчанию (default gateway) для внешнего прямого IP адреса, назначенного данному серверу. Если серверу не назначен прямой внешний IP адрес, по данному ключу возвращается пустая строка. |
| ext_ip | string | Прямой внешний IP адрес, назначенный данному виртуальному серверу. Если серверу не назначен прямой внешний IP адрес, по данному ключу возвращается пустая строка. |
| ext_mac | string | MAC адрес виртуального сетевого интерфейса, подключенного к прямому внешнему (публичному) IP адресу. Если серверу не назначен прямой внешний IP адрес, по данному ключу возвращается пустая строка. |
| ext_netmask | int | Маска подсети для прямого внешнего IP адреса. Если серверу не назначен прямой внешний IP адрес, по данному ключу возвращается пустая строка. |
| id | int | Уникальный цифровой идентификатор виртуального сервера в платформе DECS. |
| int_ip | string | Внутренний IP адрес виртуального сервера (внутри виртуального сетевого сегмента, в котором был создан данный сервер). |

| Ключ | Тип данных | Описание |
|---|---|---|
| name | string | Имя виртуального сервера.<br>Имя уникально только в рамках одного и того же виртуального сетевого сегмента. |
| password | string | Пароль системного пользователя по умолчанию. |
| state | string | Состояние виртуального сервера. |
| username | string | Имя системного пользователя по умолчанию. |
| vdc_ext_ip | string | Внешний (публичный) IP адрес виртуального сетевого сегмента, к которому принадлежит данный сервер. |
| vdc_id | int | Уникальный цифровой идентификатор виртуального сетевого сегмента (VDC), к которому принадлежит данный сервер. |
| vdc_name | string | Имя виртуального сетевого сегмента, к которому принадлежит данный севере.<br>Имя уникально только в рамках одного и того же подписчика (*tenant*-а). |

# Типовые сценарии и примеры использования

## Как создать новый сервер в новом виртуальном сетевом сегменте.

Данный сценарий удобен, когда нужно создать новый виртуальный сервер, но не хочется утруждать себя проверками, существует ли целевой сетевой сегмент. Если целевой сетевой сегмент не существует, то он будет создан до того, как начнется создание виртуального сервера. Если же целевой сегмент уже существует, то модуль извлечет из контроллера необходимую информацию и поместит в этот сегмент новую виртуальную машину.

### Дано:

- DECS контроллер находится по адресу *cloud.digitalenergy.online*
- Пользователь *abstract_user* имеет доступ к подписчику *MyMainTenant* в указанном DECS контроллере.
- Oauth2 провайдер находится по адресу *sso.decs.online*
- Пользователь *abstract_user* идентифицируется в Oauth2 провайдере по известным Application ID & Application Secret, которые занесены в переменные *my_app_id* и *my_app_secret* соответственно.

### Задача:

- Авторизоваться в DECS контроллер в режиме *oauth2*.
- Создать новый виртуальный сетевой сегмент, принадлежащий подписчику *MyMainTenant*, со следующими характеристиками:
    - Имя сегмента – *MyFirstVDC* (предполагается, что у данного подписчика еще нет сегмента с таким именем).
    - Расположение – в дата-центре *site1*.
- Поместить в только что созданный виртуальный сетевой сегмент новый виртуальный сервер с именем *NewVM01*. Сервер должен иметь следующие характеристики:

- CPU – 2.
- RAM – 4096 МБ.
- Загрузочный диск – 10 ГБ.
- Образ ОС – “*Ubuntu 16.04*”.
- Описание – “My new VM created and managed with DECS VM module”

- Сохранить информацию о созданном виртуальном сервере в переменной *vm01_specs*.

```
---
- hosts: ansible_master
  tasks:
  - name: create new VM in a new private network segment
    decs_vm:
      authenticator: oauth2
      app_id: "{{ my_app_id }}"
      app_secret: "{{ my_app_secret }}"
      oauth2_url: "https://sso.decs.online"
      controller_url: "https://cloud.digitalenergy.online"
      name: NewVM01
      state: present
      cpu: 2
      ram: 4096
      boot_disk:
        size: 10
      image_name: "Ubuntu 16.04"
      tenant: "MyMainTenant"
      datacenter: site1
      vdc_name: "MyFirstVDC"
      annotation: "My new VM created and managed with DECS VM module"
    delegate_to: localhost
    register: vm01_specs
...
```

Следует иметь ввиду, что для создания нового сетевого сегмента обязательны следующие параметры: `tenant, datacenter, vdc_name`. Если хотя бы один из этих параметров не указан, а сегмент не существует, модуль вернет ошибку.

Если достоверно известно, что целевой сегмент существует, то для его идентификации будет достаточно `tenant` и `vdc_name` (либо просто указать его уникальный идентификатор `vdc_id` – см. пример в разделе «Как создать новый сервер в существующем виртуальном сетевом сегменте.»).

## Как создать новый сервер в существующем виртуальном сетевом сегменте.

Данный сценарий удобен для быстрого создания нового виртуального сервера или получения информации об уже существующем виртуальном сервере (подробнее об этом см. в разделе «Как получить и использовать информацию о созданном сервере внутри playbook»), когда известен цифровой идентификатор сетевого сегмента, где должен располагаться данный сервер.

Дано:

- DECS контроллер находится по адресу *cloud.digitalenergy.online*
- Пользователь *abstract_user* имеет доступ к подписчику *MyMainTenant* в указанном DECS контроллере.
- Oauth2 провайдер находится по адресу *sso.decs.online* и отвечает по порту *4443*
- Пользователь *abstract_user* идентифицируется в Oauth2 провайдере по известным Application ID & Application Secret, которые занесены в переменные *my_app_id* и *my_app_secret* соответственно.

Задача:

- Авторизоваться в DECS контроллер в режиме *oauth2*.
- Создать новый виртуальный сервер с именем *NewVM02*. Сервер должен иметь следующие характеристики:
    - Располагаться в сетевом сегменте с идентификатором *685* (предполагается, что данный сетевой сегмент уже существует и принадлежит подписчику *MyMainTenant*)
    - CPU – 1.
    - RAM – 2048 МБ.
    - Загрузочный диск – 20 ГБ.
    - Образ ОС – “*Ubuntu 16.04*”.
    - Описание – “Another VM created and managed with DECS VM module”.
- Сохранить информацию о созданном виртуальном сервере в переменной *vm02_specs*.

```
---
- hosts: ansible_master
  tasks:
  - name: create new VM in the pre-existing private network segment
    decs_vm:
      authenticator: oauth2
      app_id: "{{ my_app_id }}"
      app_secret: "{{ my_app_secret }}"
      oauth2_url: "https://sso.decs.online:4443"
      controller_url: "https://cloud.digitalenergy.online"
      name: NewVM02
      state: present
      cpu: 1
      ram: 2048
      boot_disk:
        size: 20
      image_name: "Ubuntu 16.04"
      vdc_id: 685
      annotation: "My new VM created and managed with DECS VM module"
    delegate_to: localhost
    register: vm02_specs
...
```

## Как получить и использовать информацию о созданном сервере внутри playbook

После успешного выполнения *task*, вызывающего модуль *decs_vm*, базовая информация о полученном виртуальном сервере может быть записана в переменную с помощью директивы `register` (см. последнюю строку в примерах из разделов, посвященных созданию нового виртуального сервера).

Обратите внимание, что даже если сервер с заданными в *task* характеристиками существовал на момент запуска модуля *decs_vm*, и никаких фактических изменений в его конфигурации не требовалось, модуль все равно возвращает информацию о текущем состоянии виртуального сервера. Содержание и формат этой информации подробно описаны в разделе «Возвращаемые значения».

### Дано:

- Первый task в playbook успешно создал новый виртуальный сервер и с помощью директивы `register` информация о нем сохранена в переменной `vm_specs`.
- На сервере настроена трансляция сетевых портов следующим образом:
  - Внешний порт – 21022.
  - Внутренний порт – 22 (протокол SSH).
  - Сетевой протокол – tcp.

### Задача:

- Внести запись о только что созданном сервере в *inventory* и запустить на нем *task*, который установит пакет *nginx*.

```
---
- hosts: ansible_master
  tasks:
  - name: obtain VM
    decs_vm:
    <<<детали опущены>>>
    register: vm_specs

  - name: add the VM to the in-memory inventory into group just_created
    add_host:
      groups: just_created
      hostname: "{{ vm_specs.vm_facts.name }}"
      ansible_host: "{{ vm_specs.vm_facts.vdc_ext_ip }}"
      ansible_port: 21022
      ansible_user: "{{ vm_specs.vm_facts.username }}"
      ansible_password: "{{ vm_specs.vm_facts.password }}"

- hosts: just_created
  tasks:
  - name: install the latest nginx and update package cache on the way
    apt:
        name: nginx
        state: latest
        update_cache: yes
...
```

## Как управлять правилами трансляции сетевых портов сервера.

### Дано:

- DECS контроллер находится по адресу *cloud.digitalenergy.online*
- Пользователь *abstract_user* имеет доступ к подписчику *MyMainTenant* в указанном DECS контроллере.
- Oauth2 провайдер находится по адресу *sso.decs.online*
- Пользователь *abstract_user* идентифицируется в Oauth2 провайдере по известным Application ID & Application Secret, которые занесены в переменные *my_app_id* и *my_app_secret* соответственно.

### Задача №1 – создание/добавление правил:

- Авторизоваться в DECS контроллер в режиме *oauth2*.
- Создать новый виртуальный сервер с именем *AppSrv01*. Сервер должен иметь следующие характеристики:
    - Располагаться в сетевом сегменте с именем MyApps (предполагается, что данный сетевой сегмент уже существует и принадлежит подписчику *MyMainTenant*)
    - CPU – 4.
    - RAM – 8192 МБ.
    - Загрузочный диск – 20 ГБ.
    - Образ ОС – “*Ubuntu 16.04*”.
    - Настройки трансляции сетевых портов:
        - Внешний порт 21022 -> внутренний порт 22, протокол tcp.
        - Внешний порт 443 -> внутренний порт 443, протокол tcp.
- Сохранить информацию о созданном виртуальном сервере в переменной *appsrv01*.

```
---
- hosts: ansible_master
  tasks:
  - name: create new VM with port forwards
    decs_vm:
      authenticator: oauth2
      app_id: "{{ my_app_id }}"
      app_secret: "{{ my_app_secret }}"
      oauth2_url: "https://sso.decs.online"
      controller_url: "https://cloud.digitalenergy.online"
      name: AppSrv01
      state: present
      cpu: 4
      ram: 8192
      boot_disk:
        size: 20
      image_name: "Ubuntu 16.04"
      port_forwards:
        - ext_port: 21022
          int_port: 22
          proto: tcp
        - ext_port: 443
```

```
          int_port: 443
          proto: tcp
      tenant: "MyMainTenant"
      vdc_name: "MyApps"
    delegate_to: localhost
    register: appsrv01
...
```

### Задача №2 – изменение/удаление правил:

- Авторизоваться в DECS контроллер в режиме *oauth2*.
- Изменить настройки трансляции сетевых портов у ранее созданного виртуального сервера с именем *AppSrv01* следующим образом:
  - Добавить правило «Внешний порт 80 -> внутренний порт 80, протокол tcp» удалить.
  - Правило «Внешний порт 21022 -> внутренний порт 22, протокол tcp» оставить без изменений.
  - Правило «Внешний порт 443 -> внутренний порт 443, протокол tcp» удалить.
- Прочие характеристики сервера оставить без изменений.
- Сохранить информацию о созданном виртуальном сервере в переменной *appsrv01*.

```
---
- hosts: ansible_master
  tasks:
  - name: create new VM with port forwards
    decs_vm:
      authenticator: oauth2
      app_id: "{{ my_app_id }}"
      app_secret: "{{ my_app_secret }}"
      oauth2_url: "https://sso.decs.online"
      controller_url: "https://cloud.digitalenergy.online"
      name: AppSrv01
      state: present
      port_forwards:
        - ext_port: 80
          int_port: 80
          proto: tcp
        - ext_port: 21022
          int_port: 22
          proto: tcp
      tenant: "MyMainTenant"
      vdc_name: "MyApps"
    delegate_to: localhost
    register: appsrv01
...
```

Как следует из логики вышеприведенного *playbook*, при вызове *decs_vm* для уже существующего виртуального сервера с настроенными ранее правилами трансляции сетевых адресов мы всякий раз задаем целевое состояние правил. Таким образом:

- Если какие-то из уже настроенных в настоящий момент правил в целевом состоянии отсутствуют, они будут удалены.
- Если какие-то из уже настроенных правил присутствуют в целевом состоянии, они будут оставлены без изменений.
- Если какие-то из правил, перечисленных в целевом состоянии, отсутствуют в текущем состоянии, то они будут созданы.

## Как управлять подключением / отключением внешнего IP адреса.

### Дано:

- DECS контроллер находится по адресу *cloud.digitalenergy.online*
- Пользователь *abstract_user* имеет доступ к подписчику *MyMainTenant* в указанном DECS контроллере.
- Oauth2 провайдер находится по адресу *sso.decs.online*
- Пользователь *abstract_user* идентифицируется в Oauth2 провайдере по известным Application ID & Application Secret, которые занесены в переменные *my_app_id* и *my_app_secret* соответственно.

### Задача №1 – подключить внешний IP адрес к серверу:

- Авторизоваться в DECS контроллер в режиме *oauth2*.
- Создать новый виртуальный сервер с именем *AppSrv02*. Сервер должен иметь следующие характеристики:
    - Располагаться в сетевом сегменте с именем MyApps (предполагается, что данный сетевой сегмент уже существует и принадлежит подписчику *MyMainTenant*).
    - CPU – 1.
    - RAM – 4096 МБ.
    - Загрузочный диск – 50 ГБ.
    - Образ ОС – “*Ubuntu 16.04*”.
    - Настройки трансляции сетевых портов:
        - Внешний порт 21022 -> внутренний порт 22, протокол tcp
- Выделить серверу внешний IP адрес из сетевого сегмента с идентификатором 42.
- Сохранить информацию о созданном виртуальном сервере в переменной *appsrv02*.

```
---
- hosts: ansible_master
  tasks:
  - name: create new VM with port forwards and direct external IP
    decs_vm:
      authenticator: oauth2
      app_id: "{{ my_app_id }}"
      app_secret: "{{ my_app_secret }}"
      oauth2_url: "https://sso.decs.online"
      controller_url: "https://cloud.digitalenergy.online"
      name: AppSrv02
      state: present
      cpu: 1
      ram: 4096
```

```
      boot_disk:
        size: 50
      image_name: "Ubuntu 16.04"
      ext_network: present
      ext_network_id: 42
      port_forwards:
        - ext_port: 21022
          int_port: 22
          proto: tcp
      tenant: "MyMainTenant"
      vdc_name: "MyApps"
    delegate_to: localhost
    register: appsrv02
...
```

После успешного выполнения данного *task* получить сетевые характеристики сервера из переменной *appsrv02* можно следующим образом:

- Внешний прямой IP адрес – `appsrv02.vm_facts.ext_ip`
- Шлюз по умолчанию – `appsrv02.vm_facts.ext_gateway`
- Маска подсети – `appsrv02.vm_facts.ext_netmask`
- MAC-адрес внешнего сетевого интерфейса – `appsrv02.vm_facts.ext_mac`

## Как включать / выключать существующий сервер.

### Дано:

- DECS контроллер находится по адресу *cloud.digitalenergy.online*
- Пользователь *abstract_user* имеет доступ к подписчику *MyMainTenant* в указанном DECS контроллере.
- Подписчик *MyMainTenant* обладает виртуальным сетевым сегментом с именем *MyApps*, в котором имеется виртуальный сервер с именем *AppSrv01*.
- Oauth2 провайдер находится по адресу *sso.decs.online*
- Пользователь *abstract_user* идентифицируется в Oauth2 провайдере по известным Application ID & Application Secret, которые занесены в переменные *my_app_id* и *my_app_secret* соответственно.

### Задача №1 – выключить виртуальный сервер:

- Авторизоваться в DECS контроллер в режиме *oauth2*.
- Выключить виртуальный сервер *AppSrv01*.
- Прочие характеристики виртуального сервера *AppSrv01* должны остаться без изменений.

```
---
- hosts: ansible_master
  tasks:
  - name: poweroff existing VM
    decs_vm:
      authenticator: oauth2
      app_id: "{{ my_app_id }}"
      app_secret: "{{ my_app_secret }}"
```

```
      oauth2_url: "https://sso.decs.online"
      controller_url: "https://cloud.digitalenergy.online"
      name: AppSrv01
      state: poweredoff
      tenant: "MyMainTenant"
      vdc_name: "MyApps"
    delegate_to: localhost
...
```

Аналогично, чтобы запустить остановленный виртуальный сервер, в параметр `state` надо выставить в `poweredon`. Работающий сервер можно перевести в состояние «приостановлен», выставив параметр `state` надо выставить в `paused`.

### Задача №2 – включить виртуальный сервер:

- Авторизоваться в DECS контроллер в режиме *oauth2*.
- Включить ранее остановленный виртуальный сервер *AppSrv01*.
- В процессе включения:
    - Гарантировать, что сервер не будет иметь прямого выделенного IP адреса.

```
---
- hosts: ansible_master
  tasks:
  - name: poweron existing VM, resize it and remove external IP (if any)
    decs_vm:
      authenticator: oauth2
      app_id: "{{ my_app_id }}"
      app_secret: "{{ my_app_secret }}"
      oauth2_url: "https://sso.decs.online"
      controller_url: "https://cloud.digitalenergy.online"
      name: AppSrv01
      ext_network: absent
      state: poweredon
      tenant: "MyMainTenant"
      vdc_name: "MyApps"
    delegate_to: localhost
...
```

Следует иметь ввиду, что определенные комбинации целевого и текущего состояния сервера не подразумевают изменения «параметров питания» сервера (например, целевое состояние *present*, а текущее – *poweredoff*). В таких ситуациях изменение параметров питания не производится, но может быть выполнена настройка сетевых параметров (правила трансляции сетевых портов, подключение/отключение прямого IP адреса) и изменение объема выделенных серверу ресурсов (CPU, RAM, объем загрузочного диска).

## Как изменить CPU, RAM и размер загрузочного диска существующего сервера.

### Дано:

- DECS контроллер находится по адресу *cloud.digitalenergy.online*

- Пользователь *abstract_user* имеет доступ к подписчику *MyMainTenant* в указанном DECS контроллере.
- Подписчик *MyMainTenant* обладает виртуальным сетевым сегментом с именем *MyApps*, в котором имеется виртуальный сервер с именем *AppSrv02*. Текущие характеристики сервера:
  - CPU – 2.
  - RAM – 4096 МБ.
  - Загрузочный диск – 50 ГБ.
- Oauth2 провайдер находится по адресу *sso.decs.online*
- Пользователь *abstract_user* идентифицируется в Oauth2 провайдере по известным Application ID & Application Secret, которые занесены в переменные *my_app_id* и *my_app_secret* соответственно.

Задача:

- Авторизоваться в DECS контроллер в режиме *oauth2*.
- Задать новые характеристики для виртуального сервера *AppSrv02*:
  - CPU – 4.
  - RAM – 8192 МБ.
  - Загрузочный диск – 80 ГБ.

```
---
- hosts: ansible_master
  tasks:
  - name: poweron existing VM, resize it and remove external IP (if any)
    decs_vm:
      authenticator: oauth2
      app_id: "{{ my_app_id }}"
      app_secret: "{{ my_app_secret }}"
      oauth2_url: "https://sso.decs.online"
      controller_url: "https://cloud.digitalenergy.online"
      name: AppSrv02
      cpu: 4
      ram: 8192
      bootdisk:
        size: 100
      state: present
      tenant: "MyMainTenant"
      vdc_name: "MyApps"
    delegate_to: localhost
...
```

Одновременно с изменением выделенных CPU, RAM и размера загрузочного диска сервера можно перенастроить правила трансляции сетевых портов и подключить/отключить прямой внешний IP адрес.

## Как удалить существующий сервер.

Дано:

- DECS контроллер находится по адресу *cloud.digitalenergy.online*

- Пользователь *abstract_user* имеет доступ к подписчику *MyMainTenant* в указанном DECS контроллере.
- Подписчик *MyMainTenant* обладает виртуальным сетевым сегментом с именем *MyApps*, в котором имеется виртуальный сервер с именем *AppSrv01*.
- Oauth2 провайдер находится по адресу *sso.decs.online*
- Пользователь *abstract_user* идентифицируется в Oauth2 провайдере по известным Application ID & Application Secret, которые занесены в переменные *my_app_id* и *my_app_secret* соответственно.

Задача:

- Авторизоваться в DECS контроллер в режиме *oauth2*.
- Удалить виртуальный сервер *AppSrv01*.

```
---
- hosts: ansible_master
  tasks:
  - name: delete existing VM
    decs_vm:
      authenticator: oauth2
      app_id: "{{ my_app_id }}"
      app_secret: "{{ my_app_secret }}"
      oauth2_url: "https://sso.decs.online"
      controller_url: "https://cloud.digitalenergy.online"
      name: AppSrv01
      state: absent
      tenant: "MyMainTenant"
      vdc_name: "MyApps"
    delegate_to: localhost
...
```

Следует иметь ввиду, что данная операция необратима и удаляет все дисковые ресурсы, которые были ассоциированы с данным сервером (как загрузочный, так и data-диски).

## Как выглядит task для разных режимов авторизации.

В разделе «Способы авторизации» описаны доступные в рамках облачной платформы DECS режимы авторизации. В данном разделе приведены примеры *task* для каждого из этих режимов.

### Режим *oauth2* с использованием Application ID & Application Secret

Данный режим задается установкой параметра `authenticator: oauth2`

В этом режиме необходимо указать параметры `app_id` и `app_secret` (либо задать соответствующие значения в переменных окружения DECS_APP_ID и DECS_APP_SECRET).

Помимо вышеуказанных параметров также необходимо задать `oauth2_url` и `controller_url`. В итоге task будет выглядеть следующим образом:

```
---
- hosts: ansible_master
  tasks:
```

```
  - name: manage VM in DECS cloud
    decs_vm:
      authenticator: oauth2
      app_id: "{{ my_app_id }}"
      app_secret: "{{ my_app_secret }}"
      oauth2_url: "https://sso.decs.online"
      controller_url: "https://cloud.digitalenergy.online"
    <<<дальнейшие детали опущены>>>
...
```

### Режим *jwt* с использованием заранее полученного JSON Web Token (JWT)

Данный режим задается установкой параметра `authenticator: jwt`

В этом режиме необходимо указать параметр `jwt` (либо задать соответствующее значение в переменной окружения DECS_JWT).

Помимо вышеуказанных параметров также необходимо задать `oauth2_url` и `controller_url`. В итоге task будет выглядеть следующим образом:

```
---
- hosts: ansible_master
  tasks:
  - name: manage VM in DECS cloud
    decs_vm:
      authenticator: jwt
      jwt: "{{ my_jwt }}"
      oauth2_url: "https://sso.decs.online"
      controller_url: "https://cloud.digitalenergy.online"
    <<<дальнейшие детали опущены>>>
...
```

Указывать `oauth2_url` в данном режиме требуется для проверки действительности предоставленного JWT на стадии инициализации модуля.

### Режим *legacy* с использованием имени пользователя и пароля

Данный режим задается установкой параметра `authenticator: legacy`

В этом режиме необходимо указать параметры `user` и `password` (либо задать соответствующие значения в переменных окружения DECS_USER и DECS_PASSWORD).

Помимо вышеуказанных параметров также необходимо задать `controller_url`. В итоге task будет выглядеть следующим образом:

```
---
- hosts: ansible_master
  tasks:
  - name: manage VM in DECS cloud
    decs_vm:
      authenticator: legacy
      user: "{{ legacy_user_name }}"
      password: "{{ legacy_user_password }}"
      controller_url: "https://cloud.digitalenergy.online"
```

```
    <<<дальнейшие детали опущены>>>
...
```