

# Ошибка "Cannot resolve principal" при попытке подключения к внешнему серверу аутентификации AD

## 1. Вопрос

Если попытка подключения к внешнему серверу аутентификации AD заканчивается неудачно и в логе есть запись вида

```
Cannot resolve principal 'ovirtadm@domain.local'
```

## 2. Проверка

Это говорит о работе службы глобального каталога не на порту **3268**, а **389**. Следует проверить сервисные записи DNS. Для этого перейдите в терминал (подключитесь по SSH) к менеджеру управления и введите (где **example.com** - адрес домена):

```
dig _ldap._tcp.gc._msdcs.example.com SRV
dig _ldap._tcp.example.com SRV
```

Корректный результат вывода команды `dig _ldap._tcp.gc._msdcs.example.com SRV`:

```
# dig _ldap._tcp.gc._msdcs.domain.local SRV

; <<>> DiG 9.11.36-RedHat-9.11.36-3.el8 <<>> _ldap._tcp.gc._msdcs.domain.local
SRV
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8991
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4000
;; QUESTION SECTION:
;_ldap._tcp.gc._msdcs.domain.local. IN    SRV

;; ANSWER SECTION:
```

```
_ldap._tcp.gc._msdcs.domain.local. 600 IN SRV      0 100 3268
myserver.domain.local.

;; ADDITIONAL SECTION:
myserver.domain.local. 3600 IN      A      172.25.1.19

;; Query time: 3 msec
;; SERVER: 172.25.1.19#53(172.25.1.19)
;; WHEN: Fri Dec 02 09:24:05 MSK 2022
;; MSG SIZE rcvd: 121
```

Корректный результат вывода команды `dig _ldap._tcp.example.com SRV`:

```
# dig _ldap._tcp.domain.local SRV

; <<>> DiG 9.11.36-RedHat-9.11.36-3.el8 <<>> _ldap._tcp.domain.local SRV
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44159
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4000
;; QUESTION SECTION:
_ldap._tcp.domain.local.      IN      SRV

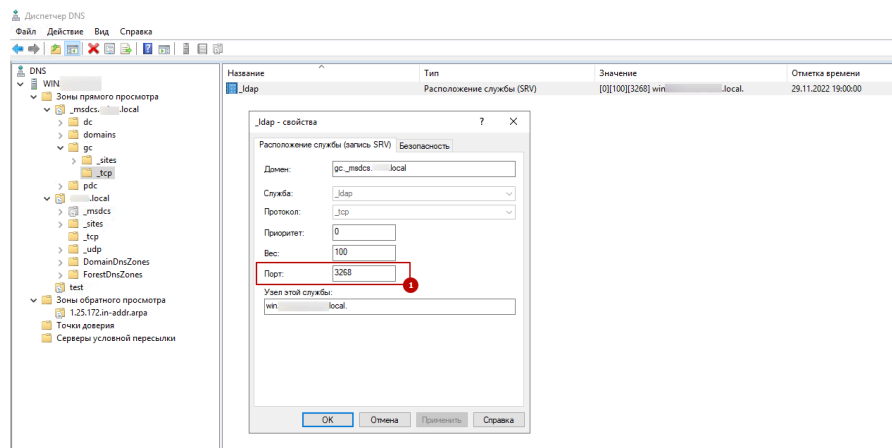
;; ANSWER SECTION:
_ldap._tcp.domain.local. 600      IN      SRV      0 100 389
myserver.domain.local.

;; ADDITIONAL SECTION:
myserver.domain.local. 3600 IN      A      172.25.1.19

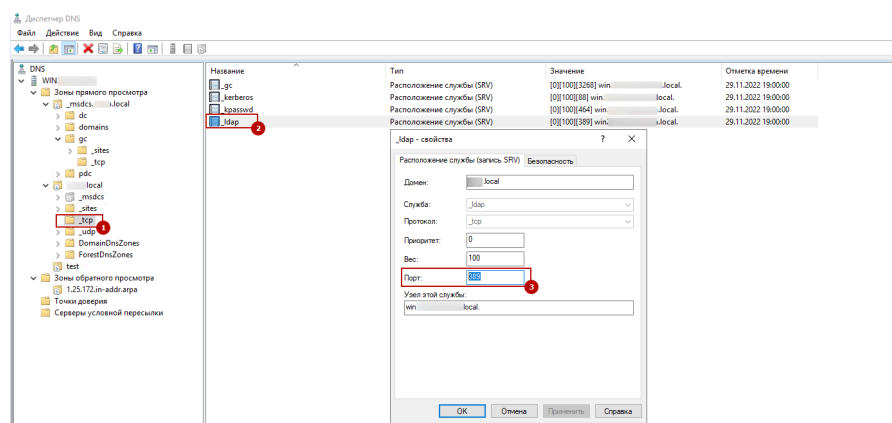
;; Query time: 2 msec
;; SERVER: 172.25.1.19#53(172.25.1.19)
;; WHEN: Fri Dec 02 09:24:42 MSK 2022
;; MSG SIZE rcvd: 111
```

В графическом виде можно проверить используемые порты можно по путям:

1. Оснастка DNS – Имя DNS сервера – Зоны прямого просмотра –  
\_msdcs.domain.name – gc – \_tcp – \_ldap.



2. Оснастка DNS – Имя DNS сервера – Зоны прямого просмотра – domain.name – \_tcp – \_ldap.



## 3. Решение

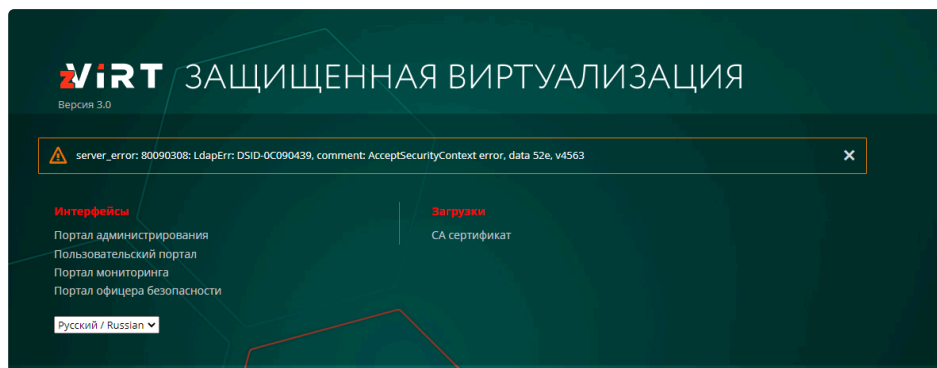
Измените значения текущих портов, на значения по умолчанию. По умолчанию LDAP работает на порту 389, GC на порту 3268.

# Ошибка "server error data 52e" при авторизации

## 1. Ошибка

При авторизации по протоколу LDAP:

```
server_error: 80090308: LdapErr: DSIS-0C090439, comment: AcceprSecurityContext error, data 52e, v4563
```



## 2. Решение

Ошибка возникает из-за неверного логина/пароля в файле `/etc/ovirt-engine/aaa/domain.properties`, где `domain` имя Вашего домена.

Необходимо ввести корректный логин в переменную `vars.user` и корректный пароль `vars.password`. Например:

```
include = <ad.properties>

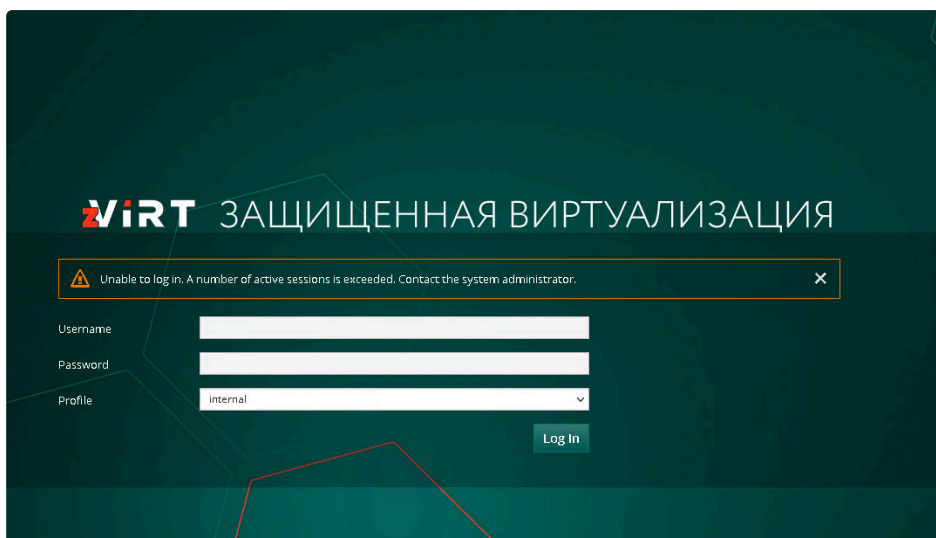
vars.domain = test.local
vars.user = CN=Qwerty Qwerty,CN=Users,DC=test,DC=local
vars.password = mypassword

pool.default.auth.simple.bindDN = ${global:vars.user}
pool.default.auth.simple.password = ${global:vars.password}
pool.default.serverset.type = srvrecord
pool.default.serverset.srvrecord.domain = ${global:vars.domain}
```

При входе в Портал администрирования  
ошибка "Unable to log in A number of  
active sessions is exceeded"

## 1. Вопрос

При входе в "Портал администрирования" ошибка "Unable to log in A numver of active sessions is exceeded"



## 2. Решение

При проверке лог файла можно обнаружить множественные подключения с УЗ **Admin**, но с разных IP адресов.

[illegible]

Необходимо рассмотреть 2 варианта:

### Вариант 1

1. Если используется локальная учетная запись, то необходимо проверить выданные ей ограничения. Перейти по пути Управление – Пользователи – Выбрать нужную учетную запись – Нажать Управление ограничениями .
2. Выставить необходимые значения количества сессий и времени сессии

Управление ограничениями пользователя

Имя пользователя

admin

Провайдер авторизации

internal-authz

Количество сессий

0

Время сессий (минуты)

0

Сохранить

Заккрыть

## Вариант 2

1. Если используется доменная учетная запись, то не рекомендуется использовать ее для работы сторонних сервисов. В данном случае невозможно полностью снять ограничения на количество сессий.

# Автозапуск виртуальных машин при перезагрузке менеджера управления (BM HostedEngine)

1. Подключитесь к Менеджеру управления через SSH или Cockpit и авторизуйтесь
2. Убедитесь, что включены репозитории **zvirt**:

```
dnf repolist all
repo id                repo name
status
zvirt-extras          zVirt extras repository
enabled ①
zvirt-main             zVirt 4.3 main repository
enabled ①
```

BASH |

① - должен быть в статусе **enabled**

3. Установите пакет **zvirt-autostart-vms**:

```
dnf install -y https://repo-zvirt.orionsoft.ru/tools/autostart-vms-1.0.0-1.147291.zvirt.el8.noarch.rpm
```

4. Отредактируйте файл конфигурации **/opt/zvirt/zvirt-autostart-vms/autostart.conf** и введите в него необходимые параметры:

```
engine_hostname: "engine.example.local" ①
engine_api_username: "admin@internal" ②
engine_api_password: "Qwerty123#" ③
login_timeout: "180" ④
autostart_pause: "10" ⑤
vms_list: ⑥
- vm1
- vm2
- vm3
- vm4
awating_dc_sec: 600 ⑦
excluded_dc: ⑧
- DC1
- DC2
```

YAML |

① - FQDN Менеджера управления.

- ② - Учетная запись пользователя в формате **user@domain** с разрешением на доступ к API.



В случае использования в zVirt 4.2 с установленным Keycloak, необходимо указать пользователя в домене **internalsso**. По умолчанию таким пользователем является **admin@zvirt@internalsso**.

- ③ - Пароль учетной записи.
- ④ - Время (в секундах), в течение которого сценарий будет пытаться подключиться к API. По умолчанию - 180.
- ⑤ - Задержка между запуском ВМ в секундах.
- ⑥ - Список ВМ, которые необходимо автоматически запускать после перезагрузки Менеджера управления.
- ⑦ - Время (в секундах), в течение которого сценарий будет ожидать перехода центра данных в состояние **Включен**.
- ⑧ - Список имен центров данных, к которым не будет применяться сценарий автозапуска. По умолчанию список пустой.



Пользователю под которым выполняется авторизация в API (указывается в параметре **engine\_api\_username**) необходимо разрешить как минимум две одновременных сессии.



# Работа с шаблонами zVirt с помощью Ansible

## Пояснение

Для использования шаблонов, а также их управлением можно использовать модули коллекции Ansible **ovirt.ovirt**. Также можно использовать коллекцию **oVirt Ansible Collection**, в которую входит роль **image\_template**, позволяющая добавлять в zVirt шаблоны из QCOW2-образов или при указании URL-ссылки на такой образ.

В примерах использования ниже предполагается, что Playbook запускается с **Менеджера управления**, информация о запуске Playbook с внешнего хоста представлена во [Введении в автоматизацию работы zVirt с помощью Ansible](#)

## 1. Создание шаблонов с помощью модуля "ovirt\_template"

Для создания шаблона из VM можно воспользоваться модулем **ovirt\_template**.

Также данный модуль позволяет удалять и переименовывать шаблоны, импортировать шаблоны и управлять их версионностью.

В качестве простого примера ниже представлен Task'и, который создаёт простой шаблон из определённой VM:

YAML | 

```
- name: Create template from VM
  ovirt_template:
    cluster: Default           # указание кластера
    storage_domain: StorageName # в каком домене хранить шаблон
    name: TemplateName        # название шаблона
    vm: rhel8                  # название VM, из которой будет создан шаблон
    description: Test          # описание шаблона
```

Ниже представлен пример Task'и, с помощью которой можно импортировать шаблон из экспорт домена.

YAML | 

```
- name:
  ovirt_template:
    auth: "{{ ovirt_auth }}"
    state: imported           # для импорта обязательно нужно указать
```

```
состояние imported
  cluster: Default
  name: Template
  export_domain: FromDomain # также нужно указать название экспорт-домена
  storage_domain: ToDomain # и имя целевого домена, в который шаблон
будет импортирован
  description: Test
```

Для удаления шаблона, нужно указать его имя и задать параметр **state: absent**.

## 2. Создание шаблонов с помощью роли "image\_template"

### 2.1. Переменные

Шаблоны и подключение к zVirt описываются с помощью переменных.

Основными переменными являются:

```
engine_fqdn:      # FQDN-имя менеджера управления
engine_user:      # Логин для авторизации
engine_password:  # Пароль для авторизации
engine_cafile:    # Переменная отвечает за корневой сертификат менеджера
управления

template_cluster: # Эта переменная указывает в каком кластере создать
шаблон
template_name:    # Данная переменная задаёт название шаблону
template_memory:  # Объём оперативной памяти
template_cpu:     # Количество ЦПУ, назначаемых VM
template_disk_size: # Объём диска VM
template_disk_storage: # Домен хранения шаблона
```

YAML | 

Некоторые переменные зависят от источника, которые будет использоваться для создания шаблона. Список всех переменных представлен в README-файле в каталоге с самой ролью.

### 2.2. Пример

Для создания шаблона из образа **QCOW2** используется переменная **qcow\_url**. В данной переменной указывается HTTP/S ссылка на QCOW2 образ или локальный путь до образа.

Пример создания шаблона из образа QCOW2 представлен ниже:

```

- name: Create a template from qcow2
  hosts: localhost
  connection: local
  gather_facts: false

  vars:
    engine_fqdn: zvirt-engine.example.com
    engine_user: ansible@internal
    engine_password: 123456
    engine_cafile: /etc/pki/ovirt-engine/ca.pem

    qcow_url: https://cloud.centos.org/centos/7/images/CentOS-7-x86_64-
GenericCloud.qcow2
    # Пример указания локального пути к образу QCOW2 ниже
    # qcow_url: file:///qcow2/CentOS7.qcow2
    template_cluster: Default
    template_name: centos7_template
    template_memory: 4GiB
    template_cpu: 2
    template_disk_size: 10GiB
    template_disk_storage: mydata

  roles:
    - image_template

```



Перед созданием шаблона из локального образа, рекомендуется назначить владельцем директории с образом и самого образа **vdsm:kvm**, для этого необходимо выполнить команду:

```
chown 36:36 -R /qcow2
```

## 3. Создание VM из шаблонов

Для создания VM из шаблона, используется модуль **ovirt\_vm** с параметрами **template**.

Простой пример Task'и, с помощью которой можно создать VM из шаблона, представлен ниже:

```

- name: Create VM
  ovirt_vm:
    state: present
    cluster: Default
    name: VMName
    template: TemplateName

```





Если нужно создать VM из шаблона с предустановленным диском, шаблон также должен быть создан с использованием RAW-формата дисков, а также указать следующие параметры: **disk\_format: raw** и **clone: true**.

Пример task'и представлен ниже:

```
- name: Create VM
  ovirt_vm:
    state: present
    cluster: Default
    name: VMName
    template: TemplateName
    storage_domain: StorageDomainName    # этот параметр позволяет указать в
    disk_format: raw                     каком домене хранения создать диски VM
    clone: true                          # с помощью двух последних параметров
                                         можно создать VM с предустановленными дисками из шаблона
```

YAML |

# Удаление устаревших LUN на хостах средствами Ansible

## 1. Пояснение

После удаления домена хранения или LUN'a непосредственно на СХД, на хостах кластера остаются неактуальные пути.

Для удаления устаревших путей с хостов, можно воспользоваться одним из 2-х способов:

1. Перезагрузка хостов после удаления домена или LUN'a на СХД;
2. Удаления путей с хостов с помощью Ansible Role `remove_stale_lun`.

## 2. Применение Ansible-роли

Для удаления оставшихся после удаления хранилища путей с помощью Ansible, необходимо скачать необходимую роль на Менеджер управления. Установка и использование ролей из коллекции **Ovirt Ansible Collection** описана во [Введении в автоматизацию работы zVirt с помощью Ansible](#).

После скачивания ролей из архива, необходимо создать playbook со следующим содержанием:

YAML | 

```
- name: oVirt remove stale LUN
  hosts: localhost
  connection: local
  gather_facts: false

  vars:
    ansible_ssh_common_args: "-o StrictHostKeyChecking=no"
    ansible_user: root
    ansible_ssh_private_key_file: /etc/pki/ovirt-engine/keys/engine_id_rsa

    engine_fqdn: zvirt.example.ru
    engine_user: admin@internal
    engine_password: 12345

    data_center: default
    lun_wwid: #!/перечислить ID через пробел!
```

```
roles:
```

```
- remove_stale_lun
```

В параметре **lun\_wwid** нужно перечислить ID LUN'ов, которые необходимо очистить с хостов.

# Динамический Ansible inventory в zVirt

## Динамический inventory в Ansible

Механизм, который позволяет Ansible динамически находить и использовать инвентаризационные данные в реальном времени из внешних источников, таких как API, базы данных и другие источники данных, а не только из INI- YAML-файлов инвентаризации.

В zVirt **динамический inventory** реализован за счёт обращение к API и сбору информации о хостах, формировании системных переменные и группировке хостов на основе общих метаданных (принадлежность к кластеру или наличие тега).

## 1. Подготовка

Для того, чтобы можно было работать с динамическим inventory в zVirt, необходимо подготовить файлы **ansible.cfg** и сам динамический inventory.



динамический inventory обязательно должен иметь название **ovirt.yaml**, так как Python проводит валидацию названия файла.

В файле **ansible.cfg** вам необходимо активировать плагин **ovirt.ovirt.ovirt** (этот плагин отвечает за работу **динамического inventory**) в секции **[inventory]**.

Пример ниже:

```
[inventory]
enable_plugins = ovirt.ovirt.ovirt, yaml, ini
```

По-умолчанию используется следующие форматы: **host\_list**, **script**, **auto**, **yaml**, **ini**, **toml**.

Если необходимо использовать иные плагины, помимо указанных в примере **ansible.cfg**, можно просто добавить их в список.

Далее нужно настроить сам inventory **ovirt.yaml**:

```
plugin: ovirt.ovirt.ovirt # Название плагина
ovirt_url: "https://zvirt.example.ru/ovirt-engine/api" # URL API zVirt
ovirt_cafile: "/root/zvirt/ca.cer" # Путь до
корневого сертификат zVirt. Файл можно положить в любую директорию, путь
рекомендуется указывать абсолютный
ovirt_username: admin@internal # Имя пользователя
```

```
с указанием пространства имён. Локальные пользователи в zVirt находятся в
пространстве @internal
ovirt_password: admin # Пароль учётной
записи
ovirt_query_filter: # Указывается
запрос для поиска хостов для Inventory
  search: 'name=*' # Запрос 'name=*'
выбирает VM с любым именем
  # search: 'name=* AND cluster=SecondCluster' # Пример запроса,
в котором выбираются VM с любым именем, состоящие в кластере SecondCluster
  case_sensitive: no # Указание
необходимости учитывать регистр
```

После создания файла, необходимо проверить работу. Для этого достаточно воспользоваться командой:

```
ansible-inventory --graph
```

BASH | 

Либо можно указать inventory файл прямо в CLI:

```
ansible-inventory -i /path/to/inventory/ovirt.yaml --graph
```

BASH | 

Команда должна обнаружить все VM:

```
@all:
  |--@ungrouped:
  | |--CentOS8Template
  | |--my-zvirt-vm
  | |--nginx
```

BASH | 

## 2. Группировка хостов

Динамический inventory позволяет также группировать хосты. Пример группировки по кластеру и тегу ниже:

```
plugin: ovirt.ovirt.ovirt
ovirt_url: "https://zvirt.example.ru/ovirt-engine/api"
ovirt_cafile: "/root/zvirt/ca.cer"
ovirt_username: admin@internal
ovirt_password: admin
ovirt_query_filter:
  search: 'name=*'
  case_sensitive: no
keyed_groups:
  - key: cluster # Группировка по ключу cluster
```

YAML | 



```
    prefix: "cluster"      # Префикс группы. В название группы будет добавлен
                             префикс, указанный здесь. Префикс и значение ключа будут разделены нижним
                             прочерком.
  - key: tags              # Группировка по ключу tags
    prefix: "tag"
```

Вывод такого inventory:

```
@all:
|--@ungrouped:
|--@cluster_Default:
|  |--CentOS8Template
|  |--my-zvirt-vm
|  |--nginx
|--@tag_my_zvirt_tag:
|  |--my-zvirt-vm
|--@tag_nginx:
|  |--nginx
```

BASH |

Все виртуальные машины попали в группу **cluster\_Default**, VM **nginx** попала в группу **tag\_nginx**, а VM **my-zvirt-vm** попала в группу **tag\_my\_zvirt\_tag**.

### 3. Добавление переменной "ansible\_host"

Чтобы хостам в inventory автоматически назначался **ansible\_host** IP-адресом, можно воспользоваться **compose** в inventory-файле.

Для этого создайте следующий inventory:

```
plugin: ovirt.ovirt.ovirt
ovirt_url: "https://zvirt.example.ru/ovirt-engine/api"
ovirt_cafile: "/root/zvirt/ca.cer"
ovirt_username: admin@internal
ovirt_password: admin
ovirt_query_filter:
  search: 'name=*'
  case_sensitive: no
keyed_groups:
  - key: cluster
    prefix: "cluster"
compose:
  ansible_host: devices["enp1s0"][0]      # Будет произведена композиция
                                           переменной ansible_host по ключу devices, в котором выбирается устройство enp1s0
                                           и выбирается первый элемент списка IP-адресов устройства
```

YAML |

Проверка inventory:

```
"my-zvirt-vm": {
  "ansible_host": "10.0.88.19",
  "cluster": "Default",
  "devices": {
    "enp1s0": [
      "10.0.88.19",
      "fe80::546f:feff:feaa:2"
    ]
  },
  "name": "my-zvirt-vm",
  "status": "up",
  "tags": [
    "my-zvirt-tag"
  ]
},
"nginx": {
  "ansible_host": "10.0.88.18",
  "cluster": "Default",
  "devices": {
    "enp1s0": [
      "10.0.88.18",
      "fe80::546f:feff:feaa:1"
    ]
  },
  "name": "nginx",
  "status": "up",
  "tags": [
    "nginx"
  ]
}
```

Вывод выше неполный, так как помимо указанных переменных, есть и иные, однако, для упрощения понимания были выбраны лишь самые важные.

В **compose** мы указывали выбрать из **devices** устройство **enp1s0** и уже из него выбрать первый элемент списка. В **devices** обоих хостов входит устройство **enp1s0**. У устройства есть список, в котором каждый элемент из себя представляет IP-адрес, в данном случае у обоих хостов у устройства **enp1s0** сначала идёт IPv4, затем IPv6. Как раз первый элемент списка мы указывали в **ovirt.yaml** и переменной **ansible\_host** присвоилось нужное значение.



Название устройств может разниться у разных ОС, перед указанием **compose** рекомендуем изучить вывод `ansible-inventory --list` без указания **compose** в inventory.

## 4. Тестирование

Для проверки был запущен модуль **ping** для различных групп, составленных с помощью **динамического inventory**

```
ansible-inventory --graph -i inventory/ovirt.yaml
```

Вывод:

```
@all:
  |--@ungrouped:
  |--@cluster_Default:
  |   |--CentOS8Template
  |   |--my-zvirt-vm
  |   |--nginx
  |--@tag_my_zvirt_tag:
  |   |--my-zvirt-vm
  |--@tag_nginx:
  |   |--nginx
[root@ansible ansible]$ ansible tag_nginx -m ping
nginx | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/libexec/platform-python"
  },
  "changed": false,
  "ping": "pong"
}
[root@ansible ansible]$ ansible tag_my_zvirt_tag -m ping
my-zvirt-vm | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/libexec/platform-python"
  },
  "changed": false,
  "ping": "pong"
}
[root@ansible ansible]$ ansible cluster_Default -m ping
nginx | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/libexec/platform-python"
  },
  "changed": false,
  "ping": "pong"
}
my-zvirt-vm | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/libexec/platform-python"
  },
  "changed": false,
  "ping": "pong"
}
CentOS8Template | SUCCESS => {
```

```
"ansible_facts": {  
  "discovered_interpreter_python": "/usr/libexec/platform-python"  
},  
"changed": false,  
"ping": "pong"  
}
```