Инструкция по установке экземпляра программного обеспечения

Полная установка

- Общие сведения
  - Общие требования
  - Исходные манифесты
- Terraform
- Ansible
  - Создание inventory
    - Список групп сервисов
    - Список хостов
    - Пример создания файла hosts.yml
    - Группа db
    - Группа caches
    - Группы fronts, batchs, storage и monitoring
  - Создание файла переменных
  - Установка компонентов
    - Установка баз данных
    - Установка сервисов кэшей
    - Установка minio
    - Установка frontend сервисов
    - Установка backend сервисов
    - Установка сервисов мониторинга

Общие сведения

В этом мануале будут описаны действия, которые необходимо произвести, для того, чтобы полностью развернуть весь перечень сервисов VKDoc.

Общие требования

Для установки сервисов, необходимо наличие, следующего списка ПО.

1. Ansible >= 5.5.0
    a. Jinja2 == `3.0.3`
2. Terraform >= v1.1.7
3. git >= 2.32.1

Исходные манифесты

Для установки on-premise решения VKDoc необходимо скачать следующие манифесты:

1. Terraform
2. Ansible

Terraform

Ansible

Создание inventory

Для начала работы со сценариями ansible, необходимо составить и описать список серверов, на которых будут установлены сервисы, для этого следует создать файл hosts.yml, в котором указать :

1. Список групп сервисов
2. Список хостов, разбитый по группам
3. IP адреса хостов

4. Опции SSH подключения к хостам

Список групп сервисов

Для каждого сервиса в VKDoc On-premise, в сценариях ansible существуют заранее подготовленные группы, которые определяют, на каком хосте будет установлен тот или иной сервис.

Для соотнесения группы и сервиса, следует воспользоваться следующей таблицей:

> Группы отмеченные нижним подчеркиванием и курсивом (*группа*) могут быть предопределены и имеют значение по умолчанию, которое имеет значение master_hrbox_{{ значение группы в единственном числе (dbs → master_hrbox_db) }}

Сервис	Группы
postgresql	consul_instances *dbs*
etcd	*dbs*
minio	*storage*
kafka/zookeeper	kafka zookeeper
redis	*mems*
front	*fronts*
batch	*batches*
elasticsearch/kibana/APM	*elks*
prometheus/grafana	-

> Prometheus и Grafana могут быть установлены на любой хост и не имеют отдельной группы

Список хостов

Список хостов, IP адреса и опции SSH подключения описываются для каждого хоста отдельно, в формате yaml.

> Подробнее, о способах подключения ansible к хосту, можно прочитать тут или с помощью команд
>
> ```
> ansible-doc -t connection -l #
> ansible-doc -t connection < connection_plugin > #
> ```

Пример описания хоста:

```
< hostname >: #
        ansible_host: < IP > # IP
        ansible_user: < ssh_user > #  ,  ansible        SSH
    ansible_ssh_private_key_file: < path_to_ssh_private_key > #    ssh ,    ansible
        ansible_port: 22 #
```

Пример создания файла hosts.yml

Ниже будет приведен пример создания hosts.yml файла для определения структуры компонентов VKDoc On-Premise

Более подробно, о составлении "инвентаря" можно прочитать тут

Именование групп и хостов в данном примере является абстрактным и может быть переопредено, за исключением групп, явно указанных в таблице

```
---
all: #   ,
  children:
    db: # ,
      hosts:
        db1:
          ansible_host: 10.0.64.1
        db2:
          ansible_host:        10.0.64.2
        db3:
          ansible_host:        10.0.64.3
        etcd1:
          ansible_host: 10.0.64.4
        etcd2:
          ansible_host: 10.0.64.5
        etcd3:
          ansible_host: 10.0.64.6
      children:
        consul_instances: # ,      consul, patroni  postgresql
          hosts:
            db1:
              ansible_host: 10.0.64.1
            db2:
              ansible_host: 10.0.64.2
            db3:
              ansible_host: 10.0.64.3

    storage: # ,      S3
      hosts:
        minio:
          ansible_host: 10.0.64.7
    caches:   # ,
      hosts:  #  ,     (!)
        cache1:
          ansible_host: 10.0.64.8
        cache2:
          ansible_host: 10.0.64.9
        cache3:
          ansible_host: 10.0.64.10
        cache4:
          ansible_host: 10.0.64.11
      children:
        kafka: #  ,     Kafka
          hosts:
            cache1:
              ansible_host: 10.0.64.8
            cache2:
              ansible_host: 10.0.64.9
            cache3:
              ansible_host: 10.0.64.10
        zookeeper: #  ,      Zookeeper,   Kafka
          hosts:
            cache1:
              ansible_host: 10.0.64.8
            cache2:
              ansible_host: 10.0.64.9
            cache3:
              ansible_host: 10.0.64.10
    fronts: # ,      frontend
```

```
    hosts:
      front1:
        ansible_host: 10.0.64.12
      front2:
        ansible_host: 10.0.64.13
  batchs: # ,      backend
    hosts:
      batch1:
         ansible_host: 10.0.64.14
      batch2:
        ansible_host: 10.0.64.15
  monitoring: # ,       (prometheus, grafana, elasticsearch, kibana, APM)
    hosts:
      monitoring1:
        ansible_host: 10.0.64.16
              monitoring2:
        ansible_host: 10.0.64.17
```

После создания файла hosts.yml, необходимо убедится, что инвентарь собран правильно, сделать это можно следующий командой :

```
ansible-inventory -i hosts.yml --list
```

Если все сделано верно, то вывод команды будет примерно такой :

```
{
    "_meta": {
        "hostvars": { #
            "batch1": {
                "ansible_host": "10.0.64.14"
            },
            "batch2": {
                "ansible_host": "10.0.64.15"
            },
            "cache1": {
                "ansible_host": "10.0.64.8"
            },
            "cache2": {
                "ansible_host": "10.0.64.9"
            },
            "cache3": {
                "ansible_host": "10.0.64.10"
            },
            "cache4": {
                "ansible_host": "10.0.64.11"
            },
            "db1": {
                "ansible_host": "10.0.64.1"
            },
            "db2": {
                "ansible_host": "10.0.64.2"
            },
            "db3": {
                "ansible_host": "10.0.64.3"
            },
            "etcd1": {
                "ansible_host": "10.0.64.4"
            },
            "etcd2": {
                "ansible_host": "10.0.64.5"
            },
            "etcd3": {
                "ansible_host": "10.0.64.6"
            },
            "front1": {
                "ansible_host": "10.0.64.12"
            },
            "front2": {
                "ansible_host": "10.0.64.13"
```

```
        },
        "minio": {
            "ansible_host": "10.0.64.7"
        },
                    "monitoring2": {
            "ansible_host": "10.0.64.17"
        },
        "monitoring1": {
            "ansible_host": "10.0.64.16"
        }
    }
},
"all": { #
    "children": [ #
        "batchs",
        "caches",
        "db",
        "fronts",
        "monitoring",
        "storage",
        "ungrouped"
    ]
},
"batchs": {
    "hosts": [ #  ,   "batchs"
        "batch1",
        "batch2"
    ]
},
"caches": {
    "children": [ #   ,  - "caches"
        "kafka",
        "zookeeper"
    ],
    "hosts": [  #  ,   "caches"
        "cache1",
        "cache2",
        "cache3",
        "cache4"
    ]
},
"consul_instances": {
    "hosts": [  #  ,   "consul_instances",      "db"
        "db1",
        "db2",
        "db3"
    ]
},
"db": {
    "children": [ #   ,  - "caches"
        "consul_instances"
    ],
    "hosts": [  #  ,   "db"
        "db1",
        "db2",
        "db3",
        "etcd1",
        "etcd2",
        "etcd3"
    ]
},
"fronts": {
    "hosts": [  #  ,   "fronts"
        "front1",
        "front2"
    ]
},
"kafka": {
    "hosts": [  #  ,   "kafka",      "caches"
        "cache1",
        "cache2",
```

```
            "cache3"
        ]
    },
    "monitoring": {
        "hosts": [  #  ,   "monitoring"
                          "monitoring1",
            "monitoring2"
        ]
    },
    "storage": {
        "hosts": [  #  ,   "storage"
            "minio"
        ]
    },
    "zookeeper": {
        "hosts": [  #  ,   "zookeeper",      "caches"
            "cache1",
            "cache2",
            "cache3"
        ]
    }
}
```

Разберем подробнее группировку хостов:

Группа db

Группа *db* имеет вложенную группу *consul_instances,* на серверах которой будет установлены сервисы consul, patroni и postgresql, в нее входят хосты db{1,2,3}. Подгруппа *consul_instances* требуется ролью consul(link to role).

Так же группа *db* имеет хосты, на которых будет установлен только сервис ETCD - etcd{1,2,3}.

Группа caches

Группа *caches* имеет две вложенных группы - *Kafka* и *zookeeper*, эти подгруппы определяют на каких хостах будут располагаться соответствующие сервисы, в данном случае, эти сервисы будут установлены на хосты cache{1,2,3}. Подгруппы *Kafka* и *zookeeper* трутся ролью Kafka(link to role).

Так же группа *caches* имеет хост, на котором будет установлен только сервис Redis - cache4.

Группы fronts, batchs, storage и monitoring

Группы fronts, batchs, storage и monitoring не имеют вложенных групп и являются простым перечислением хостов, на которые будут установлены соответствующие сервисы.

Создание файла переменных

В случае, если используются названия групп, которые не являются значениями по умолчанию, необходимо переопределить переменные, отвечающие за поиск хостов по группам, для этого необходимо создать отдельный файл в формате yml, назовем его values.yml, со следующим содержимым:

> Названия групп, соответствуют примеру и должны быть переопределены реальными названиями

```
fronts: 'fronts'
batchs: 'batchs'
dbs: 'db'
mems: 'caches'
storages: 'storage'
elks: 'monitoring'
```

Этот файл будет использоваться при вызове ролей ansible в дальнейшем.

Установка компонентов

Подробнее о ansible playbooks можно узнать тут

Для установки компонентов VKDoc On-premise, необходимо использовать playbook.yml(link to playbook), находящийся в корневой папке репозитория ansible.

Разберем его подробнее:

```
---
- hosts: "{{ dbs | default('hrbox_db', true) }}" #   ,     ansible ,
  become: true #
  roles: #
    - role: selinux
    - role: postgresql
    - role: etcd
    - role: prometheus #  prometheus,   ,     prometheus exporters
      vars: #        prometheus
        COMPONENTS:
          - name: node_exporter
            install: true
          - name: postgres_exporter
            install: true
            opts: {
              value: "DATA_SOURCE_NAME='postgresql://postgres@localhost:5432/?
sslmode=disable'\nPOSTGRES_EXPORTER_OPTS='--extend.query-path=/etc/prometheus/postgres_exporter_queries.yaml'",
              rewrite: true
            }
          - name: haproxy_exporter
            install: true
            opts: {
              value: HAPROXY_EXPORTER_OPTS='--haproxy.scrape-uri=http://localhost:9999/haproxy_stats;csv',
              rewrite: true
            }
          - name: etcd_exporter
            install: true
  tags: # ,    playbook
    - never
    - db
    - aio # ,    (!)

- hosts: "{{ mems | default('hrbox_mem', true) }}" #   ,     ansible ,
  become: true #
  pre_tasks: #   ,      ansible
    - name: Include vars
      include_vars:
        dir: vars/hrbox_mem
  roles: #
    - role: selinux
    - role: redis
    - role: kafka
    - role: #  prometheus,   ,     prometheus exporters
              vars: #        prometheus
        COMPONENTS:
          - name: node_exporter
            install: true
          - name: redis_exporter
            install: true
            opts: {
              value: REDIS_EXPORTER_OPTS='-redis.addr 0.0.0.0:6901',
              rewrite: true
            }
          - name: haproxy_exporter
            install: true
            opts: {
              value: HAPROXY_EXPORTER_OPTS='--haproxy.scrape-uri=http://localhost:9999/haproxy_stats;csv',
```

```
            rewrite: true
          }
        - name: kafka_exporter
          install: true
          opts: {
            value: 'KAFKA_EXPORTER_OPTS="--kafka.server=localhost:9092"',
            rewrite: true
          }
  tags:
    - never
    - mem
    - aio # ,    (!)

- hosts: "{{ storages | default('hrbox_storage', true) }}" #   ,     ansible ,
  become: true #
  pre_tasks: #   ,      ansible
    - name: Include vars
      include_vars:
        dir: vars/hrbox_storage
  roles: #
    - role: storage
    - role: rsyslog
    - role: prometheus
      vars:
        COMPONENTS:
          - name: node_exporter
            install: true
          - name: minio_exporter
            install: true
  tags:
    - never
    - storage
    - aio # ,    (!)

- hosts: "{{ fronts | default('hrbox_front', true) }}" #   ,     ansible ,
  become: true #
  pre_tasks: #   ,      ansible
    - name: Include vars
      include_vars:
        dir: vars/hrbox_front
  roles: #
    - role: selinux
    - role: front
    - role: prometheus #  prometheus,   ,     prometheus exporters
      vars: #        prometheus
        COMPONENTS:
          - name: node_exporter
            install: true
          - name: haproxy_exporter
            install: true
            opts: {
              value: HAPROXY_EXPORTER_OPTS='--haproxy.scrape-uri=http://localhost:9999/haproxy_stats;csv',
              rewrite: true
            }
          - name: nginx_exporter
            install: true
            opts: {
              value: NGINX_EXPORTER_OPTS='--nginx.scrape-uri=http://localhost:80/stub_status',
              rewrite: true
            }
    - role: checker
  tags:
    - never
    - front
    - aio # ,    (!)

- hosts: "{{ batchs | default('hrbox_batch', true) }}" #   ,     ansible ,
  become: true #
  pre_tasks: #   ,      ansible
    - name: Include vars
      include_vars:
```

```
        dir: vars/hrbox_batch
  roles: #
    - role: selinux
    - role: batch
    - role: prometheus #  prometheus,   ,     prometheus exporters
      vars: #        prometheus
        COMPONENTS:
          - name: node_exporter
            install: true
          - name: haproxy_exporter
            install: true
            opts: {
              value: HAPROXY_EXPORTER_OPTS='--haproxy.scrape-uri=http://localhost:9999/haproxy_stats;csv',
              rewrite: true
            }
    - role: checker
  tags:
    - never
    - batch
    - aio # ,    (!)

- hosts: "{{ elks | default('hrbox_elk', true) }}" #   ,     ansible ,    (prometheus, grafana, kibana)
  become: true #
  roles: #
    - role: ELK
    - role: #  prometheus,   ,     prometheus exporters,        prometheus
      vars: #        prometheus
        COMPONENTS:
          - name: node_exporter
            install: true
          - name: elasticsearch_exporter
            install: true
          - name: prometheus
            install: true
            opts: {
              value: PROMETHEUS_OPTS='--config.file=/etc/prometheus/prometheus.yml
              --storage.tsdb.path=/var/lib/prometheus/data
              --web.console.libraries=/usr/share/prometheus/console_libraries
              --web.console.templates=/usr/share/prometheus/consoles
              --storage.tsdb.retention.time 48h
              --storage.tsdb.retention.size 15MB',
              rewrite: true
            }
    - role: grafana
  tags:
    - never
    - elk
    - aio # ,    (!)
```

Все нижеперечисленные команды, необходимо выполнять из корневой директории репозитория ansible

Для установки сразу всех компонентов, необходимо воспользоваться следующей командой

```
ansible-playbook -i hosts.yml playbook.yml --tags aio --extra-vars "@values.yml"
```

Установка баз данных

Более подробно про ansible роли для баз данных, можно почитать тут:

- postgresql
- etcd

Для применения ansible ролей баз данных, для заранее определенных групп хостов, необходимо воспользоваться командой:

```
ansible-playbook -i hosts.yml playbook.yml --tags db --extra-vars "@values.yml"
```

Установка сервисов кэшей

Более подробно про ansible роли для сервисов кэшей , можно почитать тут:

- kafka
- redis

Для применения ansible ролей, которые установят сервисы кэшей, для заранее определенных групп хостов, необходимо воспользоваться командой:

```
ansible-playbook -i hosts.yml playbook.yml --tags mem --skip-tags always --extra-vars "@values.yml"
```

Установка minio

Более подробно про ansible роли для S3 , можно почитать тут:

- rsyslog
- minio

Для применения ansible ролей, которые установят сервисов хранения данных, для заранее определенных групп хостов, необходимо воспользоваться командой:

```
ansible-playbook -i hosts.yml playbook.yml --tags storage --skip-tags always --extra-vars "@values.yml"
```

Установка frontend сервисов

Более подробно про ansible роли для frontend сервисов , можно почитать тут

Для применения ansible ролей для frontend сервисов , для заранее определенных групп хостов, необходимо воспользоваться командой:

```
ansible-playbook -i hosts.yml playbook.yml --tags front --skip-tags always --extra-vars "@values.yml"
```

Установка backend сервисов

Более подробно про ansible роли для backend сервисов , можно почитать тут

Для применения ansible ролей для backend сервисов , для заранее определенных групп хостов, необходимо воспользоваться командой:

```
ansible-playbook -i hosts.yml playbook.yml --tags batch --skip-tags always --extra-vars "@values.yml"
```

Установка сервисов мониторинга

Более подробно про ansible роли для frontend сервисов , можно почитать тут:

- elk
- prometheus
- grafana

Для применения ansible ролей для сервисов мониторинга , для заранее определенных групп хостов, необходимо воспользоваться командой:

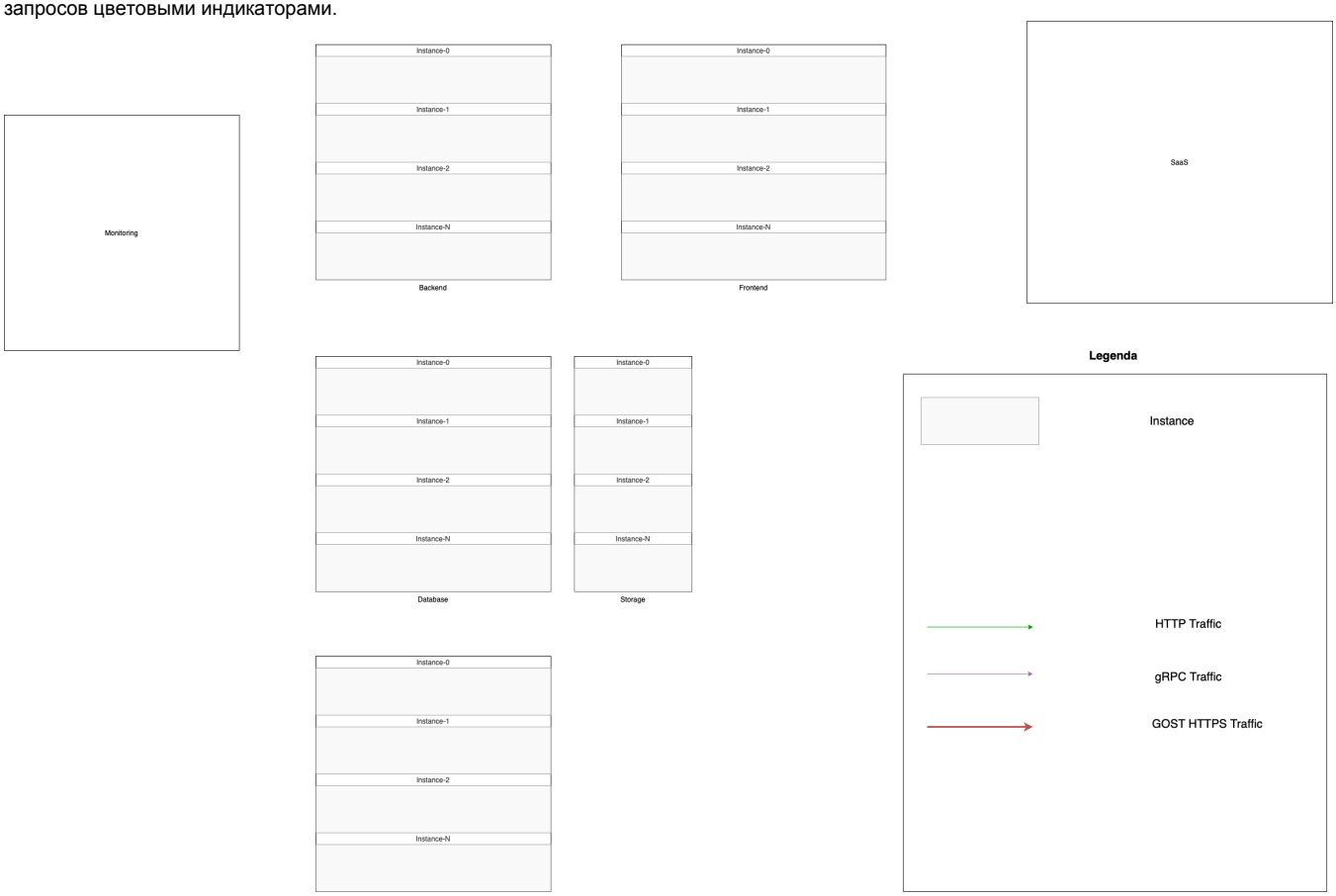
```
ansible-playbook -i hosts.yml playbook.yml --tags elk --skip-tags always --extra-vars "@values.yml"
```

Архитектура

- Общая схема архитектуры
  - Как пользоваться схемой?
- ГОСТ шифрование

Общая схема архитектуры

На схеме ниже, отображены все компоненты решения On-Premise, связи сервисов, их порты и пути передачи данных, разделенные по типам запросов цветовыми индикаторами.



Как пользоваться схемой?

На схеме есть несколько уровней (layers):

- main - главный уровень, отвечает за отображение компонентов и разбиение их на instances
- services - отвечает за отображение frontend и backend сервисов
- apps ports - отображает порты всех сервисов и утилит имеющихся в решении On-Premise
- mem - отвечает за отображение сервисов и утилит на instances Mem, т.е кэшей
- storage - отображает структуру сервисов предназначенных для хранения данных
- databases - отвечает за отображение структуры instances отвечающих за базы данных
- monitoring - отображение все exporters используемые в решении On-premise, а так же описывает структуру instance ELK, которая отвечает за мониторинг системы и сбор логов
- monitoring ports - отображает порты всех exporters используемых для мониторинга всех компонентов решения
- networking - отвечает за отображение сетевых связей и направленности потоков данных в решении

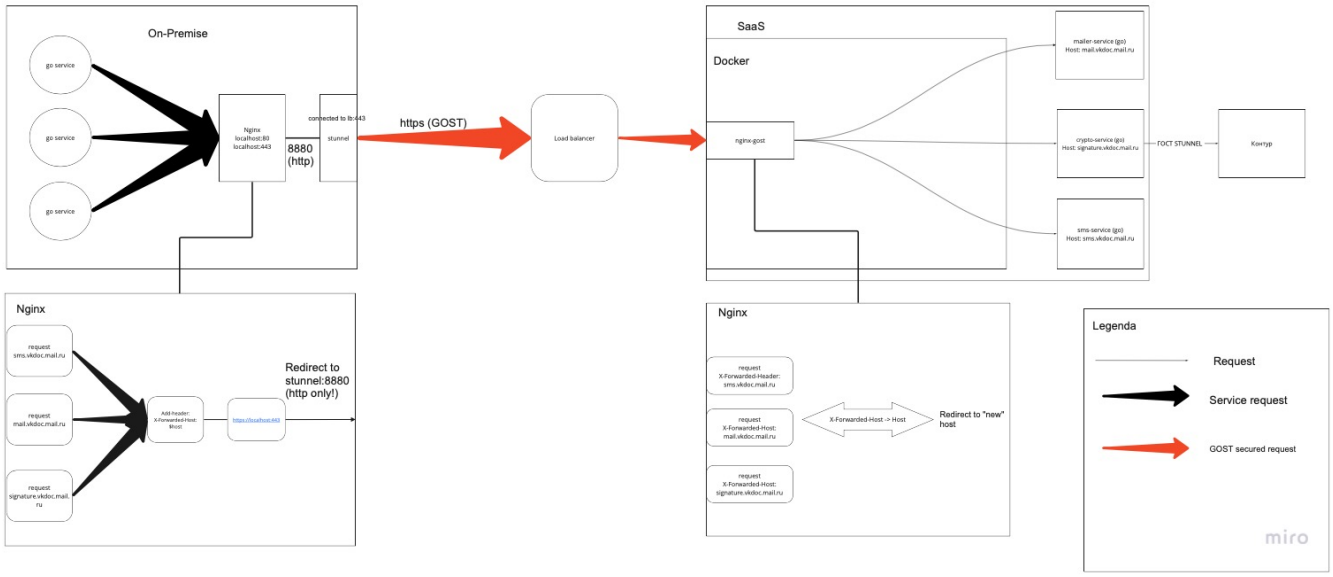
При работе со схемой необходимо переключать и комбинировать уровни, для получения необходимой информации об архитектуре решения on-premise.

Так, например, для просмотра потоков передачи данных между системами мониторинга и exporters, необходимо одновременно отобразить уровни monitoring, networking и main.

ГОСТ шифрование

При работе приложений, чувствительные данные передаются по защищенному туннелю, в зашифрованном по ГОСТ виде.

Принцип взаимодействия с туннелем приведен на рисунке ниже.



Batch services

- Общие сведения
- Terraform
- Ansible
    - Переменные роли
    - Использование роли
        - Тестирование роли
        - Вызов роли

Общие сведения

Batch services - это основные backend сервисы VKDoc.

Batch services используют Go, как основной язык и Postgresql как основную базу данных.

Terraform

> Использование terraform module требует наличия Terraform версии, не ниже `0.13,` `Openstack`

Для разворачивания инфраструктуры при помощи terraform необходимо воспользоваться terraform manifest.
Подробную инструкцию, об использовании terraform module, можно найти тут.

После того как terraform manifest будет складирован на локальную машину необходимо создать необходимые ресурсы, следующими командами :

```
terraform init #   , terraform
terraform apply --auto-approve #  terraform module       resources
```

> флаг --auto-approve в команде terraform apply --auto-approve, необходим для того, чтобы автоматически подтвердить создание инфраструктуры

Ansible

> Ansible role
>
> molecule

Ansible role front имеет зависимости от ролей:

- selinux
- vkdoc-app
- haproxy

Переменные роли

postgres_user:

- Тип - строка
- Значение по умолчанию - vkdoc_back
- Определяет имя пользователя Postgresql, который будет использован сервисом при работе.

postgres_users:

- Тип - Список объектов
- Значение по умолчанию - "{{ vault_secrets.postgres_users }}"
- Значение переменной скрыто в зашифрованном файле. Определяет пароли от пользователей postgresql.

postgres_db:

- Тип - строка
- Значение по умолчанию - vkdoc
- Определяет название базы данных postgresql для работы сервисов.

postgres_schemas:

- Тип - список строк
- Значение по умолчанию -

```
[
  "core_service",
  "crypto_service",
  "file_service",
  "fixture_service",
  "onecvk_service",
  "public_api_service",
  "sap_service",
  "web_api_service",
  "admin_service",
  "document_service"
]
```

- Определяет список схем базы данных postgresql, которые будут созданы при установке сервисов.

configs_path:

- Тип - строка
- Значение по умолчанию - /home/sites
- Определяет директорию, в которых сервисы будут искать свои конфигурационные файлы.

> Под каждый сервис создается своя собственная директория. В переменной указывается общий путь.

logs_path:

- Тип - строка
- Значение по умолчанию - /logs
- Определяет общую директорию для хранения логов сервисов.

vkdoc_core_service_config:

- Тип - объект
- Значение по умолчанию -

```
notify_mailru:
  base_url: https://notifygw.mail.ru/
  notify:
    application: mrg_vkdoc
    signature_key: "{{ vault_secrets.notify_signature_key }}"
    service: vkdoc_registration
    test_mode: false
  verify:
   application: mrg_vkdoc
   signature_key: "{{ vault_secrets.notify_signature_key }}"
   service: vkdoc_registration
   test_mode: false
   verification_code_length: 6
  mailer_mailru:
    url: https://mailer.mail.ru
    login: VKDocTest
    secret: "{{ vault_secrets.mailer_secret }}"
  invite_sender:
    url: "https://{{ basehost }}/auth/invite?token={code}"
    email_letter_id: 12439
    email_letter_id_phone_change: 13575
  links:
    front: https://{{ basehost }}
  letters:
    ids:
      default: 12563
    emails:
      dismiss_groups: hrbox-officer-dismissed@lists.vk.team
  commands:
    active_node_report:
      fallback_email: hrtek_no_hr_email@lists.vk.team
    cancel_node_report:
      fallback_email: hrtek_no_hr_email@lists.vk.team
    rate_limit:
      disable: "{{ core_service_rate_limit | default('true') }}"
```

- Определяет параметры конфигурации сервиса Core.

vkdoc_crypto_service_config:

- Тип - объект
- Значение по умолчанию -

```
kontur_crypto_api:
  auth:
    url: "http://hrbox-kontur-mock/crypto"
    host_header: ""
    secret_key: "secret_crypto_key"
  kontur_kcr_api:
    auth:
      url: http://hrbox-kontur-mock/kcr
      secret_key: "secret_kcr_key"
```

- Определяет параметры конфигурации сервиса Crypto.

vkdoc_file_service_config:

- Тип - объект
- Значение по умолчанию -

```
converter:
  tfk_unoconv:
    tfk_client:
      url: http://localhost:3001
```

- Определяет параметры конфигурации file сервиса.

vkdoc_web_api_service_config:

- Тип - объект
- Значение по умолчанию -

```
vk_team_auth:
  frontend_auth_result_url: "https://{{ basehost }}/auth/result?auth_code=%s"
  client:
    client_id: "{{ vault_secrets.vk_team_auth.client_id }}"
    client_secret: "{{ vault_secrets.vk_team_auth.client_secret }}"
    redirect_uri: "https://web-api.{{ basehost }}/api/v1/vk_team/auth_form/callback"
hasher:
  salt: "{{ vault_secrets.web_api_service_hasher_salt }}"
```

- Определяет параметры конфигурации сервиса web api.

vkdoc_sap_service_config:

- Тип - объект
- Значение по умолчанию -

```
encryptor:
  key: "{{ vault_secrets.sap_encryptor_key }}"
sap_mappings:
  head_of_first_personnel_department_group: 10
  head_of_second_personnel_department_group: 144
  first_personnel_department_group: 11
  second_personnel_department_group: 145
  personnel_department_sign_role: 6
  sign_role: 7
```

- Определяет параметры конфигурации сервиса sap.

vkdoc_onecvk_service_config:

- Тип - объект
- Значение по умолчанию -

  ```
  encryptor:
    key: "{{ vault_secrets.onecvk_encryptor_key }}"
  hasher:
    salt: "{{ vault_secrets.onecvk_hasher_salt }}"
  keychain:
    user_token_key: "{{ vault_secrets.onecvk_keychain_user_token_key }}"
  ```

- Определяет параметры конфигурации сервиса onecvk.

vkdoc_public_api_service_config:

- Тип - объект
- Значение по умолчанию -

```
hasher:
  salt: "{{ vault_secrets.public_api_hasher_salt }}"
object_storage:
  minio:
    health_check:
      buckets:
        - vkdoc-documents
```

- Определяет параметры конфигурации сервиса public api.

vkdoc_document_service_config:

- Тип - объект
- Значение по умолчанию - {}
- Определяет параметры конфигурации document сервиса.

vkdoc_fixture_service_config:

- Тип - объект
- Значение по умолчанию - {}
- Определяет параметры конфигурации fixture сервиса.

syslog:

- Тип - объект
- Значение по умолчанию -

```
address: '127.0.0.1'
port: 514
proto: udp
```

- Определяет адрес, порт и протокол для настройки сервиса Rsyslog.

Переменные типа *_version являются общими для нескольких ролей и находятся в файле group_vars/all/vkdoc_versions.yml

static_version:

- Тип - строка
- Значение по умолчанию - 22.22.5-1.el7
- Определяет версию установки static сервиса.

vkdoc_core_service_version:

- Тип - строка
- Значение по умолчанию - 22.22.7-1.el7
- Определяет версию установки Core сервиса.

vkdoc_crypto_service_version:

- Тип - строка

- Значение по умолчанию - 22.22.4-1.el7
- Определяет версию установки Crypto сервиса.

vkdoc_document_service_version:

- Тип - строка
- Значение по умолчанию - 22.22.2-1.el7
- Определяет версию установки Document сервиса.

vkdoc_file_service_version:

- Тип - строка
- Значение по умолчанию - 22.19.2-1.el7
- Определяет версию установки File сервиса.

vkdoc_onecvk_service_version:

- Тип - строка
- Значение по умолчанию - 22.23.0-1.el7
- Определяет версию установки Onvecvk сервиса.

vkdoc_public_api_service_version:

- Тип - строка
- Значение по умолчанию - 22.23.0-1.el7
- Определяет версию установки сервиса Public Api.

vkdoc_sap_service_version:

- Тип - строка
- Значение по умолчанию - 22.22.2-1.el7
- Определяет версию установки Sap сервиса.

vkdoc_web_api_service_version:

- Тип - строка
- Значение по умолчанию - 22.23.0-1.el7
- Определяет версию установки Web Api сервиса.

vkdoc_fixture_service_version:

- Тип - строка
- Значение по умолчанию - 22.28.3-1.el7
- Определяет версию установки Fixture сервиса.

apps:

- Тип - список объектов
- Значение по умолчанию :

```
- package: vkdoc_core_service_worker
  version: "{{ vkdoc_core_service_version }}"
  port: 6502
  config: "{{ vkdoc_core_service_config }}"
- package: vkdoc_crypto_service_worker
  version: "{{ vkdoc_crypto_service_version }}"
  port: 6503
  config: "{{ vkdoc_crypto_service_config }}"
- package: vkdoc_sap_service_worker
  version: "{{ vkdoc_sap_service_version }}"
  port: 6505
  config: "{{ vkdoc_sap_service_config }}"
- package: vkdoc_onecvk_service_worker
  version: "{{ vkdoc_onecvk_service_version }}"
  port: 6506
  config: "{{ vkdoc_onecvk_service_config }}"
```

- Определяет список сервисов для установки, их порты, версии и конфигурации.

other_packages:

- Тип - список строк
- Значение по умолчанию - ["mayday_migrate_command-20.30.0-1.el7.x86_64"]
- Определяет список вспомогательных пакетов, которые будут доставлены при установке сервисов.

rsyslog_max_message_size:

- Тип - строка
- Значение по умолчанию - 128k
- Определяет максимальный размер лога, который сервис Rsyslog способен получить.

Использование роли

Тестирование роли

Для использования molecule необходимо наличие python модуля molecule>=3.6.1 и molecule-openstack>=0.3

Для тестирования ansible роли, используется molecule. Для запуска тестирования необходимо, находясь в корневой папке проекта ansible, вызвать следующую команду:

```
molecule converge -s vkdoc_apps
```

В результате, будет создан тестовый instance, запустится ansible role и тесты, которые проверят работоспособность сервисов VKDoc.

После завершения тестов, сделает вызвать команду, которая удалит тестовый instance :

```
molecule destroy -s vkdoc_apps
```

Вызов роли

Для использования Ansible roles необходимо наличие установленной Ansible

Для запуска Ansible role необходимо воспользоваться следующей командой :

```
ansible-playbook -i <inventory_file_path> playbook.yml --tags batch --skip-tags always
```

ETCD

- Общие сведения
- Terraform
- Ansible
    - Переменные роли
    - Использование роли
        - Тестирование роли
        - Вызов роли

Общие сведения

официальный сайт

ETCD - база данных типа ключ-значение. Используется для service discovery сервисов (в основном go-based). В значениях хранятся версии используемых сервисов, а также grpc endpoint'ы в (hostname:port или ip:port).

Terraform

Использование terraform module требует наличия Terraform версии, не ниже `0.13,` `Openstack`

Для разворачивания инфраструктуры при помощи terraform необходимо воспользоваться terraform manifest.
Подробную инструкцию, об использовании terraform module, можно найти тут.

После того как terraform manifest будет складирован на локальную машину необходимо создать необходимые ресурсы, следующими командами :

```
terraform init #   , terraform
terraform apply --auto-approve #  terraform module        resources
```

флаг --auto-approve в команде terraform apply --auto-approve, необходим для того, чтобы автоматически подтвердить создание инфраструктуры

Ansible

Ansible role

molecule

Переменные роли

etcd_api_version:

- Тип - строка
- Значение по умолчанию - "3"
- Версия API который использует кластер.

etcd_bin_dir:

- Тип - строка
- Значение по умолчанию - "/usr/local/bin"
- Определяет директорию в которой располагаются утилиты сервиса.

etcd_cert_dir:

- Тип - строка
- Значение по умолчанию - "/etc/ssl/etcd"
- Определяет директорию в которой находятся SSL сертификаты, необходимые для работы кластера и общения между нодами.

etcd_client_port:

- Тип - строка
- Значение по умолчанию - "2379"
- Определяет порт, по которому к кластеру будут обращаться клиентские приложения

etcd_cluster_setup:

- Тип - булевое
- Значение по умолчанию - true
- Определяет, будет ли текущая инсталляция представлена в виде кластера или single-node базой данных.

etcd_config_dir:

- Тип - строка
- Значение по умолчанию - "/etc/etcd"
- Определяет директорию к которой будет обращаться сервис, в поисках конфигурационных файлов

etcd_data_dir:

- Тип - строка
- Значение по умолчанию - "/var/lib/etcd"
- Определяет директорию, в которой будут находится данные, используемые сервисом

etcd_initial_cluster_token:

- Тип - строка
- Значение по умолчанию - "etcd-hrbox"
- Определяет токен, который будет использоваться кластером, в качестве идентификатора, для подключения отдельных нод.

etcd_install_packages:

- Тип - булевое
- Значение по умолчанию - true
- Определяет необходимость установки утилит etcd и etcdctl

etcd_interface:

- Тип - строка
- Значение по умолчанию - "eth0"
- Определяет сетевой интерфейс на котором будет работать сервис. Необходим для корректного отбора IP адресов нод.

etcd_version:

- Тип - строка
- Значение по умолчанию - "v3.5.0"
- Определяет версию сервиса.

etcd_peer_port:

- Тип - строка
- Значение по умолчанию - "2380"
- Определяет порт для общения нод внутри кластера.

etcd_ipv4_address:

- Тип - строка
- Значение по умолчанию - "{{ hostvars[inventory_hostname]['ansible_facts'][etcd_interface]['ipv4']['address'] }}"
- Вычисляемое значение, отбирает IP адрес, на котором необходимо разместить сервис.

Использование роли

Тестирование роли

Для использования molecule необходимо наличие python модуля molecule>=3.6.1 и molecule-openstack>=0.3

Для тестирования ansible роли, используется molecule. Для запуска тестирования необходимо, находясь в корневой папке проекта ansible, вызвать следующую команду:

```
molecule converge -s etcd
```

В результате, будет создан тестовый instance, запустится ansible role и тесты, которые проверят работоспособность кластера ETCD.

После завершения тестов, сделает вызвать команду, которая удалит тестовый instance :

```
molecule destroy -s etcd
```

Вызов роли

Для использования Ansible roles необходимо наличие установленной Ansible

Для запуска Ansible role необходимо воспользоваться следующей командой :

```
ansible-playbook -i <inventory_file_path> playbook.yml --tags db --skip-tags always
```

Front services

- Общие сведения
- Terraform
- Ansible
    - Переменные роли
    - Использование роли
        - Тестирование роли
        - Вызов роли

Общие сведения

Front services - основные frontend сервисы VKDoc.

Front services используют php, php-fpm, nodejs, docker и postgresql.

Terraform

> Использование terraform module требует наличия Terraform версии, не ниже `0.13,` `Openstack`

Для разворачивания инфраструктуры при помощи terraform необходимо воспользоваться terraform manifest.
Подробную инструкцию, об использовании terraform module, можно найти тут.

После того как terraform manifest будет складирован на локальную машину необходимо создать необходимые ресурсы, следующими командами :

```
terraform init #   , terraform
terraform apply --auto-approve #  terraform module       resources
```

> флаг --auto-approve в команде terraform apply --auto-approve, необходим для того, чтобы автоматически подтвердить создание инфраструктуры

Ansible

> Ansible role
>
> molecule

Ansible role front имеет зависимости от ролей:

- default
- nginx
- haproxy

Переменные роли

admin_service_packages:

Тип - список строк

Значение по умолчанию -

```
[
  "vkdoc_admin_service_admin-22.22.4-1.el7.noarch"
]
```

Определяет список пакетов для установки Admin сервиса.

other_packages:

- Тип - список строк
- Значение по умолчанию -

```
[
  "mayday_migrate_command-20.30.0-1.el7.x86_64"
]
```

- Определяет список вспомогательных пакетов для установки.

php_extensions:

- Тип - список строк
- Значение по умолчанию -

```
[
  "php81-php-bcmath.x86_64",
  "php81-php-cli.x86_64",
  "php81-php-gd.x86_64",
  "php81-php-intl.x86_64",
  "php81-php-mbstring.x86_64",
  "php81-php-mysqlnd.x86_64",
  "php81-php-opcache.x86_64",
  "php81-php-pdo.x86_64",
  "php81-php-pecl-grpc.x86_64",
  "php81-php-pecl-igbinary.x86_64",
  "php81-php-pecl-igbinary-devel.x86_64",
  "php81-php-pecl-msgpack.x86_64",
  "php81-php-pecl-msgpack-devel.x86_64",
  "php81-php-pecl-protobuf.x86_64",
  "php81-php-pecl-rdkafka5.x86_64",
  "php81-php-pecl-redis5.x86_64",
  "php81-php-pgsql.x86_64",
  "php81-php-process.x86_64",
  "php81-php-soap.x86_64",
  "php81-php-pecl-xmldiff-devel.x86_64",
  "php81-php-pecl-xmldiff.x86_64",
  "php81-php-pecl-xmlrpc.x86_64",
  "php81-php-xml.x86_64",
]
```

- Определяет список расширений для языка Php, необходимые, для корректной работы сервисов.

vkdoc_core_service_config:

- Тип - объект
- Значение по умолчанию -

```
notify_mailru:
  base_url: https://notifygw.mail.ru/
  notify:
    application: mrg_vkdoc
    signature_key: "{{ vault_secrets.notify_signature_key }}"
    service: vkdoc_registration
    test_mode: false
  verify:
   application: mrg_vkdoc
   signature_key: "{{ vault_secrets.notify_signature_key }}"
   service: vkdoc_registration
   test_mode: false
   verification_code_length: 6
  mailer_mailru:
    url: https://mailer.mail.ru
    login: VKDocTest
    secret: "{{ vault_secrets.mailer_secret }}"
  invite_sender:
    url: "https://{{ basehost }}/auth/invite?token={code}"
    email_letter_id: 12439
    email_letter_id_phone_change: 13575
  links:
    front: https://{{ basehost }}
  letters:
    ids:
      default: 12563
    emails:
      dismiss_groups: hrbox-officer-dismissed@lists.vk.team
  commands:
    active_node_report:
      fallback_email: hrtek_no_hr_email@lists.vk.team
    cancel_node_report:
      fallback_email: hrtek_no_hr_email@lists.vk.team
    rate_limit:
      disable: "{{ core_service_rate_limit | default('true') }}"
```

- Определяет параметры конфигурации сервиса Core.

vkdoc_crypto_service_config:

- Тип - объект
- Значение по умолчанию -

```
kontur_crypto_api:
  auth:
    url: "http://hrbox-kontur-mock/crypto"
    host_header: ""
    secret_key: "secret_crypto_key"
  kontur_kcr_api:
    auth:
      url: http://hrbox-kontur-mock/kcr
      secret_key: "secret_kcr_key"
```

- Определяет параметры конфигурации сервиса Crypto.

vkdoc_file_service_config:

- Тип - объект
- Значение по умолчанию -

```
converter:
  tfk_unoconv:
    tfk_client:
      url: http://localhost:3001
```

- Определяет параметры конфигурации file сервиса.

vkdoc_web_api_service_config:

- Тип - объект
- Значение по умолчанию -

```
vk_team_auth:
  frontend_auth_result_url: "https://{{ basehost }}/auth/result?auth_code=%s"
  client:
    client_id: "{{ vault_secrets.vk_team_auth.client_id }}"
    client_secret: "{{ vault_secrets.vk_team_auth.client_secret }}"
    redirect_uri: "https://web-api.{{ basehost }}/api/v1/vk_team/auth_form/callback"
hasher:
  salt: "{{ vault_secrets.web_api_service_hasher_salt }}"
```

- Определяет параметры конфигурации сервиса web api.

vkdoc_sap_service_config:

- Тип - объект
- Значение по умолчанию -

```
encryptor:
  key: "{{ vault_secrets.sap_encryptor_key }}"
sap_mappings:
  head_of_first_personnel_department_group: 10
  head_of_second_personnel_department_group: 144
  first_personnel_department_group: 11
  second_personnel_department_group: 145
  personnel_department_sign_role: 6
  sign_role: 7
```

- Определяет параметры конфигурации сервиса sap.

vkdoc_onecvk_service_config:

- Тип - объект
- Значение по умолчанию -

```
encryptor:
  key: "{{ vault_secrets.onecvk_encryptor_key }}"
hasher:
  salt: "{{ vault_secrets.onecvk_hasher_salt }}"
keychain:
  user_token_key: "{{ vault_secrets.onecvk_keychain_user_token_key }}"
```

- Определяет параметры конфигурации сервиса onecvk.

vkdoc_public_api_service_config:

- Тип - объект
- Значение по умолчанию -

```
hasher:
  salt: "{{ vault_secrets.public_api_hasher_salt }}"
object_storage:
  minio:
    health_check:
      buckets:
        - vkdoc-documents
```

- Определяет параметры конфигурации сервиса public api.

vkdoc_document_service_config:

- Тип - объект
- Значение по умолчанию - {}
- Определяет параметры конфигурации document сервиса.

vkdoc_fixture_service_config:

- Тип - объект
- Значение по умолчанию - {}
- Определяет параметры конфигурации fixture сервиса.

Переменные типа *_version являются общими для нескольких ролей и находятся в файле group_vars/all/vkdoc_versions.yml

static_version:

- Тип - строка
- Значение по умолчанию - 22.22.5-1.el7
- Определяет версию установки static сервиса.

vkdoc_core_service_version:

- Тип - строка
- Значение по умолчанию - 22.22.7-1.el7
- Определяет версию установки Core сервиса.

vkdoc_crypto_service_version:

- Тип - строка
- Значение по умолчанию - 22.22.4-1.el7
- Определяет версию установки Crypto сервиса.

vkdoc_document_service_version:

- Тип - строка
- Значение по умолчанию - 22.22.2-1.el7
- Определяет версию установки Document сервиса.

vkdoc_file_service_version:

- Тип - строка
- Значение по умолчанию - 22.19.2-1.el7
- Определяет версию установки File сервиса.

vkdoc_onecvk_service_version:

- Тип - строка
- Значение по умолчанию - 22.23.0-1.el7
- Определяет версию установки Onvecvk сервиса.

vkdoc_public_api_service_version:

- Тип - строка
- Значение по умолчанию - 22.23.0-1.el7
- Определяет версию установки сервиса Public Api.

vkdoc_sap_service_version:

- Тип - строка
- Значение по умолчанию - 22.22.2-1.el7
- Определяет версию установки Sap сервиса.

vkdoc_web_api_service_version:

- Тип - строка
- Значение по умолчанию - 22.23.0-1.el7
- Определяет версию установки Web Api сервиса.

vkdoc_fixture_service_version:

- Тип - строка
- Значение по умолчанию - 22.28.3-1.el7
- Определяет версию установки Fixture сервиса.

apps:

- Тип - список объектов
- Значение по умолчанию -

```
- package: vkdoc_core_service_worker
  version: "{{ vkdoc_core_service_version }}"
  port: 6502
  config: "{{ vkdoc_core_service_config }}"
- package: vkdoc_crypto_service_worker
  version: "{{ vkdoc_crypto_service_version }}"
  port: 6503
  config: "{{ vkdoc_crypto_service_config }}"
- package: vkdoc_sap_service_worker
  version: "{{ vkdoc_sap_service_version }}"
  port: 6505
  config: "{{ vkdoc_sap_service_config }}"
- package: vkdoc_onecvk_service_worker
  version: "{{ vkdoc_onecvk_service_version }}"
  port: 6506
  config: "{{ vkdoc_onecvk_service_config }}"
```

- Определяет список конфигураций backend сервисов.

Использование роли

Тестирование роли

Для использования molecule необходимо наличие python модуля molecule>=3.6.1 и molecule-openstack>=0.3

Для тестирования ansible роли, используется molecule. Для запуска тестирования необходимо, находясь в корневой папке проекта ansible, вызвать следующую команду:

```
molecule converge -s vkdoc_apps
```

В результате, будет создан тестовый instance, запустится ansible role и тесты, которые проверят работоспособность сервисов VKDoc.

После завершения тестов, сделает вызвать команду, которая удалит тестовый instance :

```
molecule destroy -s vkdoc_apps
```

Вызов роли

Для использования Ansible roles необходимо наличие установленной Ansible

Для запуска Ansible role необходимо воспользоваться следующей командой :

```
ansible-playbook -i <inventory_file_path> playbook.yml --tags front --skip-tags always
```

Grafana

- Общие сведения
- Ansible
  - Переменные роли
  - Использование роли
    - Тестирование роли
    - Вызов роли

Общие сведения

официальный сайт

Grafana - сервис помогающий визуализировать метрики, собранные при помощи exporters и агрегированные в сервисе Prometheus.

Ansible

Ansible role

molecule

Переменные роли

GRAFANA_VERSION:

- Тип - строка
- Значение по умолчанию - 8.4.4
- Определяет версию сервиса, которая будет установлена

GRAFANA_PORT:

- Тип - число
- Значение по умолчанию - '{{ 3100 if IS_ALL_IN_ONE | default("false", true) else 3000 }}'
- Определяет порт который будет слушать сервис

Использование роли

Тестирование роли

Для использования molecule необходимо наличие python модуля molecule>=3.6.1 и molecule-openstack>=0.3

Для тестирования ansible роли, используется molecule. Для запуска тестирования необходимо, находясь в корневой папке проекта ansible, вызвать следующую команду:

```
molecule converge -s grafana
```

В результате, будет создан тестовый instance, запустится ansible role и тесты, которые проверят работоспособность сервиса Grafana.

После завершения тестов, сделает вызвать команду, которая удалит тестовый instance :

```
molecule destroy -s grafana
```

Вызов роли

Для использования Ansible roles необходимо наличие установленной Ansible

Для запуска Ansible role необходимо воспользоваться следующей командой :

```
ansible-playbook -i <inventory_file_path> playbook.yml --tags elk --skip-tags always
```

Haproxy

- Общие сведения
- Ansible
    - Переменные роли
    - Использование роли
        - Вызов роли

Общие сведения

Официальный сайт

Haproxy используется для балансировки, и проксирования TCP и HTTP запросов.

Ansible

Ansible role

Переменные роли

_haproxy_etc_prefix:

- Тип - строка
- Значение по умолчанию - /etc
- Определяет корневую директорию для установки сервиса

_haproxy_config_dir:

- Тип - строка
- Значение по умолчанию - "{{ _haproxy_etc_prefix }}/haproxy"
- Определяет директорию в которой сервис будет искать конфигурационные файлы

_haproxy_config_file:

- Тип - строка
- Значение по умолчанию - "{{ _haproxy_config_dir }}/haproxy.cfg"
- Определяет местонахождение главного конфигурационного файла

_haproxy_package_name:

- Тип - строка
- Значение по умолчанию - haproxy
- Определяет название сервиса в rpm пакете

_haproxy_extra_packages:

- Тип - список строк
- Значение по умолчанию - ["socat"]
- Определяет список вспомогательных пакетов

haproxy_repo:

- Тип - строка
- Значение по умолчанию - "*"
- Определяет yum репозитории, которые будут использоваться для установки haproxy

haproxy_config_dir:

- Тип - строка
- Значение по умолчанию - "{{ _haproxy_config_dir }}"
- Определяет директорию в которой хранятся конфигурационные файлы сервиса

haproxy_config_file:

- Тип - строка
- Значение по умолчанию - "{{ _haproxy_config_dir }}/haproxy.cfg"
- Определяет путь до главного конфигурационного файла сервиса.

haproxy_package_name:

- Тип - строка
- Значение по умолчанию - "{{ _haproxy_package_name }}"
- Определяет название пакета, который будет установлен

haproxy_extra_packages:

- Тип - строка
- Значение по умолчанию - "{{ _haproxy_extra_packages }}"
- Определяет список вспомогательный пакетов для установки

haproxy_manage_config:

- Тип - булевое
- Значение по умолчанию - true
- Определяет необходима ли конфигурация сервиса, после его установки

_haproxy_ssl_options:

- Тип - строка
- Значение по умолчанию - 'no-sslv3 no-tls-tickets'
- Определяет опции конфигурации SSL

_haproxy_ssl_ciphers:

- Тип - строка
- Значение по умолчанию - 'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-RSA-AES256-SHA:ECDHE-ECDSA-DES-CBC3-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:DES-CBC3-SHA:!DSS'
- Определяет список алгоритмов или инструкций, которые помогают создавать защищенное соединение

haproxy_global:

- Тип - объект
- Значение по умолчанию -

```
log:
  - address: /dev/log
    facility: local0
  - address: /dev/log
    facility: local1
    level: notice
chroot: /var/lib/haproxy
user: haproxy
group: haproxy
daemon: true
ssl_default_bind_options: '{{ _haproxy_ssl_options }}'
ssl_default_bind_ciphers: '{{ _haproxy_ssl_ciphers }}'
ssl_default_server_options: '{{ _haproxy_ssl_options }}'
ssl_default_server_ciphers: '{{ _haproxy_ssl_ciphers }}'
tune:
  ssl:
    default-dh-param: 2048
```

- Определяет список глобальных настроек сервиса

haproxy_defaults:

- Тип - объект
- Значение по умолчанию -

```
mode: http
log:
  - address: /dev/log
    facility: local1
    level: notice
timeout:
  - param: 'connect'
    value: '5000ms'
  - param: 'client'
    value: '50000ms'
  - param: 'server'
    value: '50000ms'
options:
  - httpclose
  - forwardfor except 127.0.0.0/8
  - redispatch
  - abortonclose
  - httplog
  - dontlognull
errorfile:
  - code: 400
    file: /usr/share/haproxy/400.http
  - code: 403
    file: /usr/share/haproxy/403.http
  - code: 408
    file: /usr/share/haproxy/408.http
  - code: 500
    file: /usr/share/haproxy/500.http
  - code: 502
    file: /usr/share/haproxy/502.http
  - code: 503
    file: /usr/share/haproxy/503.http
  - code: 504
    file: /usr/share/haproxy/504.http
```

- Определяет опций и настроек по умолчанию для сервиса

haproxy_backends, haproxy_frontends, haproxy_listen - список опций, которые зависимые сервисы переопределяют индивидуально. Пример конфигурации, можно посмотреть тут

haproxy_backends:

- Тип - список объектов
- Значение по умолчанию - []
- Определяет список backend сервисов и их конфигурации

пример переменной конфигурации haproxy_backends для роли postgresql :

```
psql_haproxy_backends:
  - name: patroni
    mode: http
    timeout:
      - param: connect
        value: 10s
      - param: server
        value: 1m
    servers:
      - name: patroni
        ip: 127.0.0.1
        port: 8009
  - name: vkdoc_db_master
    mode: tcp
    balance: leastconn
    http_check_expect:
      - status 200
    options:
      - "httpchk GET /master"
    timeout:
      - param: connect
        value: 10s
      - param: server
        value: 20m
    servers: "{{ postgresql_list }}"
```

Для переопределения значений роли, в файле meta/main.yml необходимо, в списке dependencies задать параметр vars :

```
dependencies:
  - role: default
  - role: consul
  - role: haproxy
    vars:
      haproxy_frontends: '{{ psql_haproxy_frontends }}'
      haproxy_backends: '{{ psql_haproxy_backends }}'
      haproxy_defaults: '{{ psql_haproxy_defaults }}'
```

haproxy_frontends:

- Тип - список объектов
- Значение по умолчанию - []
- Определяет список frontend сервисов и их конфигурацию

пример переменной конфигурации haproxy_frontends для роли postgresql :

```
psql_haproxy_defaults:
  mode: tcp
  options:
    - dontlognull
    - redispatch
  retries: 3
  timeout:
    - param: queue
      value: 1m
    - param: connect
      value: 10s
    - param: client
      value: 20m
    - param: server
      value: 20m
    - param: check
      value: 10s
  maxconn: 2000
  log:
    - address: global
```

Для переопределения значений роли, в файле meta/main.yml необходимо, в списке dependencies задать параметр vars :

```
dependencies:
  - role: default
  - role: consul
  - role: haproxy
    vars:
      haproxy_frontends: '{{ psql_haproxy_frontends }}'
      haproxy_backends: '{{ psql_haproxy_backends }}'
      haproxy_defaults: '{{ psql_haproxy_defaults }}'
```

haproxy_listen:

- Тип - список объектов
- Значение по умолчанию - []
- Определяет список прослушиваемых портов, портов проксирования и методы проверок health-check

пример переменной конфигурации haproxy_listen для роли batch :

```
batch_haproxy_listen:
  - name: infra-redis-queue
    bind: [ ":6500" ]
    mode: tcp
    log: global
    balance: roundrobin
    options:
      - tcp-check
    tcp_check:
      - 'connect'
      - 'send PING\r\n'
      - 'expect string +PONG'
      - 'send info\ replication\r\n'
      - 'expect string role:master'
      - 'send info\ replication\r\n'
      - 'expect string slave0'
      - 'send QUIT\r\n'
      - 'expect string +OK'
    timeout:
     - param: connect
       value: 3s
     - param: client
       value: 120m
     - param: server
       value: 120m
    servers: "{{ redis_list }}"
  - name: etcd_infra
    bind: [ ":2479" ]
    mode: tcp
    log: global
    timeout:
     - param: connect
       value: 3s
     - param: client
       value: 120m
     - param: server
       value: 120m
    options:
     - httpchk GET /health HTTP/1.0
    http_check_expect:
     - string "true"
    servers: "{{ etcd_list }}"
```

Для переопределения значений роли, в файле meta/main.yml необходимо, в списке dependencies задать параметр vars :

```
dependencies:
  - role: selinux
  - role: vkdoc-app
  - role: default
  - role: haproxy
    vars:
      haproxy_listen: '{{ batch_haproxy_listen }}'
      haproxy_frontends: '{{ batch_haproxy_frontends }}'
      haproxy_backends: '{{ batch_haproxy_backends }}'
```

haproxy_userlists:

- Тип - список объектов
- Значение по умолчанию - []
- Определяет список пользователей и групп пользователей

Использование роли

Вызов роли

> Для использования Ansible roles необходимо наличие установленной Ansible

Для запуска Ansible role необходимо воспользоваться следующей командой :

```
ansible-playbook -i <inventory_file_path> <playbook_path>.yaml
```

Kafka

- Общие сведения
- Terraform
- Ansible
    - Переменные ролей
    - Использование Ansible roles
        - Тестирование роли
        - Вызов роли

Общие сведения

> Официальный сайт

Kafka Apache — распределенная система обмена сообщениями между серверными приложениями в режиме реального времени. Написана на языках Java и Scala.

Кратко архитектуру системы сообщений можно охарактеризовать следующим образом:

- **распределенность —** отдельные узлы системы располагаются на нескольких серверах. Это обеспечивает ей высокую отказоустойчивость;
- **масштабируемость —** систему можно наращивать за счет простого добавления новых узлов.

В архитектуре Kafka Apache ключевыми являются концепции:

- **продюсер (producer) —** приложение или процесс, генерирующий и посылающий данные;
- **потребитель (consumer) —** приложение или процесс, который принимает сгенерированное продюсером сообщение;
- **сообщение —** пакет данных, необходимый для совершения какой-либо операции (например, авторизации, оформления покупки или подписки);
- **брокер —** узел (диспетчер) передачи сообщения от процесса-продюсера приложению-потребителю;
- **топик (тема) —** виртуальное хранилище сообщений (журнал записей) одинакового или похожего содержания, из которого приложение-потребитель извлекает необходимую ему информацию.

Terraform

> Использование terraform module требует наличия Terraform версии, не ниже `0.13,` `Openstack`

Для разворачивания инфраструктуры при помощи terraform необходимо воспользоваться terraform manifest.
Подробную инструкцию, об использовании terraform module, можно найти тут.

После того как terraform manifest будет складирован на локальную машину необходимо создать необходимые ресурсы, следующими командами :

```
terraform init #   , terraform
terraform apply --auto-approve #  terraform module              resources
```

> флаг --auto-approve в команде terraform apply --auto-approve, необходим для того, чтобы автоматически подтвердить создание инфраструктуры

Ansible

После создания инфраструктуры, необходимо воспользоваться Ansible role, предварительно создав playbook в котором указать ansible_groups и названия ролей, которые будут использованы для этих групп.

Пример playbook для установки и конфигурирования Apache Kafka :

```
---
- hosts: hrbox_mem
  roles:
    - role: mounting
      vars:
        MOUNT_DEST: /data
    - role: kafka
```

В представленном playbook будут использованы роли mounting, которая позволит создать и сконфигурировать physical volume из vm_custom_volume созданного при использовании terraform.

Переменная MOUNT_DEST указывает к какой папке необходимо примонтировать созданный volume.

После того как physical volume будет создан и сконфигурирован, настанет очередь роли, которая отвечает за установку и настройку Apache Kafka и Apache Zookeeper.

Переменные ролей

Для переопределения любой из переменных, можно использовать блок vars или изменить из непосредственно в файлах ролей (не рекомендуется)

файл

IS_ALL_IN_ONE :

- Тип - булиевое
- Значение по умолчанию - false
- Обязательный
- Используется для указания конфигурации, для которой запускается роль. Если значение - true, то переменной KAFKA_HEAP_OPTS будет присвоено статичное значение 2G, вне зависимости от конфигурации instance. Это сделано для того, чтобы ограничить размер java heap для instance с большим кол-вом RAM.

KAFKA_ID_FILE :

- Тип - строка
- Значение по умолчанию - /opt/kafka/broker_id.txt
- Обязательный
- Используется для определения пути файла, в который будет записан уникальный идентификатор брокера Kafka. В случае, если этот файл уже существует, то идентификатор будет взят из файла и не будет сгенерирован, что позволяет обеспечить иденпотентность.

KAFKA_VERSION :

- Тип - строка
- Значение по умолчанию - 3.2.0
- Обязательный
- Отвечает за версию Apache Kafka которая будет скачана и установлена на машинах

KAFKA_USER :

- Тип - строка
- Значение по умолчанию - kafka

- Обязательный
- Отвечает за login пользователя, которому будет принадлежать папка в которую будет установлена Apache Kafka, так же, от имени выбранного пользователя будут запускаться сервисы Apache Kafka & Zookeeper

KAFKA_GROUP :

- Тип - строка
- Значение по умолчанию - kafka
- Обязательный
- Отвечает за group пользователя, которому будет принадлежать папка в которую будет установлена Apache Kafka, так же, от group выбранного пользователя будут запускаться сервисы Apache Kafka & Zookeeper

KAFKA_HOME_DIR :

- Тип - строка
- Значение по умолчанию - /opt/kafka
- Обязательный
- Определяет в какую директорию будут установлены файлы, необходимые для работы сервисов Apache Kafka & Zookeeper

KAFKA_APP_LOGS_DIR :

- Тип - строка
- Значение по умолчанию - /var/logs/kafka
- Обязательный
- Определяет местонахождение папки, в которую будут складываться логи приложения Apache Kafka

ZOOKEEPER_DATA_LOGS_DIR :

- Тип - строка
- Значение по умолчанию - /var/log/zookeeper
- Обязательный
- Определяет размещения логов формирующихся при создании snapshots

ZOOKEEPER_DATA_DIR :

- Тип - строка
- Значение по умолчанию - /data/zookeeper
- Обязательный
- Определяет директорию в которой будут хранится snapshots данных zookeeper

KAFKA_HEAP_OPTS :

- Тип - строка, вычисляемое
- Значение по умолчанию - вычисляется в зависимости от ресурсов instance на котором разворачивается role
- Обязательный
- Отвечает за выделение количества памяти (RAM) для сервиса Apache Kafka. Значение вычисляется на основе максимально доступного кол-ва RAM деленного на 1.5 и округленного до целого. Значение используется как размер инициализированного (минимального) размера пула (-Xms), так и как максимальное размер пула (-Xmx) в Mb.

Использование Ansible roles

Тестирование роли

Для использования molecule необходимо наличие python модуля molecule>=3.6.1 и molecule-openstack>=0.3

Для тестирования ansible роли, используется molecule. Для запуска тестирования необходимо, находясь в корневой папке проекта ansible, вызвать следующую команду:

```
molecule converge -s kafka
```

В результате, будет создан тестовый instance, запустится ansible role и тесты, которые проверят работоспособность сервиса Kafka.

После завершения тестов, сделает вызвать команду, которая удалит тестовый instance :

```
molecule destroy -s kafka
```

Вызов роли

> Для использования Ansible roles необходимо наличие установленной Ansible

Для запуска Ansible role необходимо воспользоваться следующей командой :

```
ansible-playbook -i <inventory_file_path> playbook.yml --tags mem --skip-tags always
```

Minio

- Общие сведения
- Terraform
- Ansible
    - Переменные роли
    - Использование роли
        - Тестирование роли
        - Вызов роли

Общие сведения

официальный сайт

Minio - S3 хранилище, используется для хранения данных используемых сервисами.

Terraform

Использование terraform module требует наличия Terraform версии, не ниже `0.13,` `Openstack`

Для разворачивания инфраструктуры при помощи terraform необходимо воспользоваться terraform manifest.
Подробную инструкцию, об использовании terraform module, можно найти тут.

После того как terraform manifest будет складирован на локальную машину необходимо создать необходимые ресурсы, следующими командами :

```
terraform init #   , terraform
terraform apply --auto-approve #  terraform module       resources
```

флаг --auto-approve в команде terraform apply --auto-approve, необходим для того, чтобы автоматически подтвердить создание инфраструктуры

Ansible

Ansible role

molecule

Переменные роли

minio_instances:

- Тип - список объектов
- Значение по умолчанию - [
  {name: vkdoc, access_key: "{{ minio_keys.access }}", secret_key: "{{ minio_keys.secret }}", port: 9000, datadir: /minio/vkdoc, configdir: /minio /conf }
  ]
- Определяет список значений для каждой конфигурации хранилища. Сюда входят:
    - Название хранилища
    - Ключ доступа к хранилищу

- Пароль от хранилища
- Порт на котором работает сервис хранилища
- Директория, в которой будут хранится объекты хранилища
- Директория, к которой сервис будет искать конфигурационные файлы

> Значения переменных minio_keys.access и minio_keys.secret находятся в зашифрованном файле и требуют пароля, для расшифровки при использовании роли.

minio_iface:

- Тип - строка
- Значение по умолчанию - 'eth0'
- Определяет сетевой интерфейс, на котором будет работать сервис. Необходим, для корректного отбора IP адресов сервисов хранилища.

minio_ipv4_address:

- Тип - строка
- Значение по умолчанию - "{{ hostvars[inventory_hostname]['ansible_facts'][minio_iface]['ipv4']['address'] }}"
- Вычисляемое значение, отбирает IP адрес, на котором работает сервис

Использование роли

Тестирование роли

> Для использования molecule необходимо наличие python модуля molecule>=3.6.1 и molecule-openstack>=0.3

Для тестирования ansible роли, используется molecule. Для запуска тестирования необходимо, находясь в корневой папке проекта ansible, вызвать следующую команду:

```
molecule converge -s storage
```

В результате, будет создан тестовый instance, запустится ansible role и тесты, которые проверят работоспособность кластера Minio.

После завершения тестов, сделает вызвать команду, которая удалит тестовый instance :

```
molecule destroy -s storage
```

Вызов роли

> Для использования Ansible roles необходимо наличие установленной Ansible

Для запуска Ansible role необходимо воспользоваться следующей командой :

```
ansible-playbook -i <inventory_file_path> playbook.yml --tags storage --skip-tags always
```

Mounting

- Общие сведения
- Ansible
  - Переменные роли
  - Использование роли
    - Вызов роли

Общие сведения

Ansible role mounting - используется для подготовке к работе volumes, которые присоединяются к instences при работе с terraform.

Ansible

Ansible role

Переменные роли

MOUN_DEST :

- Тип строка
- Значение по умолчанию - ""
- Обязательный
- Отвечает за конечную точку монтирования physical volume

`MOUNT_SRC :`

-
-
  ```
  - ""
  ```
-
-
  ```
  physical volume.    ,     device,    partitions      vd ( vdb, vdc, vde  .)
  ```

Использование роли

Вызов роли

Для использования Ansible roles необходимо наличие установленной Ansible

Для запуска Ansible role необходимо воспользоваться следующей командой :

```
ansible-playbook -i <inventory_file_path> <playbook_path>.yaml
```

Nginx

- Общие сведения
- Ansible
  - Переменные роли
  - Использование роли
    - Вызов роли

Общие сведения

Официальный сайт

Nginx используется для корректного распределения и балансировки http(s) запросов на сервер.

Ansible

Ansible role

Переменные роли

dummy_main:

- Тип - строка
- Значение по умолчанию - "0.0.0.0"
- Определяет адрес, по которому будет осуществлять прослушивание портов 443 и 80 для основных сервисов VKDoc.

dummy_admin:

- Тип- строка
- Значение по умолчанию - "0.0.0.0"
- Определяет адрес, по которому будет осуществлять прослушивание портов 443 и 80 для Admin сервиса VKDoc.

dummy_api:

- Тип - строка
- Значение по умолчанию - "0.0.0.0"
- Определяет адрес, по которому будет осуществлять прослушивание портов 443 и 80 для сервисов API VKDoc.

dummy_bo:

- Тип - строка
- Значение по умолчанию - "0.0.0.0"
- Определяет адрес, по которому будет осуществлять прослушивание портов 443 и 80 для сервисов Stunnel VKDoc.

main_cert:

- Тип - зашифрованная строка
- Значение по умолчанию - openssl сертификат в зашифрованном виде
- Имеет значение сертификата, который будет применен на сервере, для предоставления возможности использования SSL по адресу vkdoc.mail.ru.

main_key:

- Тип - зашифрованная строка
- Значение по умолчанию - openssl ключ в зашифрованном виде
- Имеет значение ключа от сертификата, который будет использоваться для проверки подлинности SSL сертификата для vkdoc.mail.ru, используемого на сервере.

dhparam:

- Тип - зашифрованная строка
- Значение по умолчанию - openssl сертификат в зашифрованном виде
- Имеет значение сертификата, используемого для домена *.vkdoc.ru

Использование роли

Вызов роли

Для использования Ansible roles необходимо наличие установленной Ansible

Для запуска Ansible role необходимо воспользоваться следующей командой :

```
ansible-playbook -i <inventory_file_path> <playbook_path>.yaml
```

Postgresql

- Общие сведения
- Terraform
- Ansible
    - Переменные роли
    - Использование роли
        - Тестирование роли
        - Вызов роли

Общие сведения

официальный сайт

Postgresql - реляционная база данных, которая используется большинством сервисов VKDoc.

Для достижения отказустойчивости, в конфигурации On-premise используется Patroni cluster в связке с Hashicorp Consul.

Terraform

Использование terraform module требует наличия Terraform версии, не ниже `0.13,` `Openstack`

Для разворачивания инфраструктуры при помощи terraform необходимо воспользоваться terraform manifest.
Подробную инструкцию, об использовании terraform module, можно найти тут.

После того как terraform manifest будет складирован на локальную машину необходимо создать необходимые ресурсы, следующими командами :

```
terraform init #   , terraform
terraform apply --auto-approve #  terraform module       resources
```

флаг --auto-approve в команде terraform apply --auto-approve, необходим для того, чтобы автоматически подтвердить создание инфраструктуры

Ansible

Ansible role

molecule

Переменные роли

postgresql_enablerepo:

- Тип - строка
- Значение по умолчанию - ""
- При установке базы данных можно указать, какие репозитории rpm использовать.

postgresql_restarted_state:

- Тип - строка
- Значение по умолчанию - "restarted"
- Определяет какой метод будет использоваться для перезагрузки сервиса.

> Подробнее о статусах сервисов в ansible module, можно найти тут.

postgresql_python_library:

- Тип - строка
- Значение по умолчанию -python3-psycopg2
- Определяет список вспомогательных pyhon библиотек, для работы с Postgresql.

postgresql_user:

- Тип - строка
- Значение по умолчанию - postgres
- Имя пользователя для которого будут создаваться директории, файлы и запускаться сервисы.

postgresql_group:

- Тип - строка
- Значение по умолчанию - postgres
- Название группы для которой будут создаваться директории, файлы и запускаться сервисы.

postgresql_bin_path:

- Тип - строка
- Значение по умолчанию - "/usr/pgsql-11/bin"
- Определяет корневую директорию, для использования утилит сервиса.

postgresql_data_dir:

- Тип - строка
- Значение по умолчанию - "/var/lib/pgsql/11/data"
- Определяет директорию, в которой будут хранится файлы создаваемые базой данных.

postgresql_auth_method:

- Тип - строка
- Значение по умолчанию - "{{ ansible_fips | ternary('scram-sha-256', 'md5') }}"
- Определяет метод шифромавания паролей. Значение вычисляемое, если на хост соответствует FIPS 140-2, то при установке сервиса, будет выбран способ - scram-sha-256.

> Подробнее о способах шифрования паролей авторизации, можно узнать тут.

postgresql_unix_socket_directories:

- Тип - список строк
- Значение по умолчанию - "["/var/run/postgresql"]
- Определяет путь до Unix socket, который использует сервис.

postgresql_service_state:

- Тип - строка
- Значение по умолчанию - started
- Определяет состояние к которому должен прийти сервис, при его старте.

postgresql_service_enabled:

- Тип - булевое
- Значение по умолчанию - true
- Определяет, должен ли сервис стартовать автоматически, например, при перезагрузке хоста.

postgresql_global_config_options:

- Тип - список объектов
- Значение по умолчанию - [
- option: unix_socket_directories
  value: '{{ postgresql_unix_socket_directories | join(",") }}'
-]
- Определяет значения, которые будут записаны в файл конфигурации - postgresql.conf.

> Подробнее о вариантах и конфигурациях файла postgresql.conf, можно узнать тут

postgresql_hba_entries:

- Тип - спиок объектов
- Значение по умолчанию - [
  {type: local, database: all, user: postgres, auth_method: peer},
  {type: local, database: all, user: all, auth_method: peer},
  {type: host, database: all, user: all, address: '127.0.0.1/32', auth_method: "{{ postgresql_auth_method }}"},
  {type: host, database: all, user: all, address: '::1/128', auth_method: "{{ postgresql_auth_method }}"}
  ]
- Определяет набор правил, для конфигурации файла pg_hba.conf.

> Подробнее о конфигурации файла pg_hba.conf, можно узнать тут.

postgresql_databases:

- Тип - список объектов
- Значение по умолчанию - []
- Определяет список баз данных, которые будут созданы автоматически, после установки Postgresql.
  EX:

```
- name: exampledb # required; the rest are optional
  lc_collate: # defaults to 'en_US.UTF-8'
  lc_ctype: # defaults to 'en_US.UTF-8'
  encoding: # defaults to 'UTF-8'
  template: # defaults to 'template0'
  login_host: # defaults to 'localhost'
  login_password: # defaults to not set
  login_user: # defaults to '{{ postgresql_user }}'
  login_unix_socket: # defaults to 1st of postgresql_unix_socket_directories
  port: # defaults to not set
  owner: # defaults to postgresql_user
  state: # defaults to 'present'
```

postgresql_users:

- Тип - список объектов
- Значение по умолчанию - []
- Определяет список пользователей, которые должны быть созданы автоматически, после установки Postgresql.
  EX:

```
- name: jdoe #required; the rest are optional
  password: # defaults to not set
  encrypted: # defaults to not set
  priv: # defaults to not set
  role_attr_flags: # defaults to not set
  db: # defaults to not set
  login_host: # defaults to 'localhost'
  login_password: # defaults to not set
  login_user: # defaults to '{{ postgresql_user }}'
  login_unix_socket: # defaults to 1st of postgresql_unix_socket_directories
  port: # defaults to not set
  state: # defaults to 'present'
```

postgres_users_no_log:

- Тип - булевое
- Значение по умолчанию - true
- Определяет, собирать ли логи пользователей базы данных.

patroni

- Тип - словарь
- Значение по-умолчанию

```
patroni:
  namespace: patroni
  scope: vkdoc_db
  name: "{{ inventory_hostname|replace('.e','.i') }}"
  consul:
    host: 127.0.0.1:8500
  restapi:
      listen: '*:8009'
      connect_address: '{{ ansible_default_ipv4.address }}:8009'
  bootstrap:
    dcs:
      ttl: 30
      loop_wait: 10
      retry_timeout: 10
      maximum_lag_on_failover: 1048576
      postgresql:
        use_pg_rewind: true
        use_slots: true
        parameters:
          archive_command: '/usr/bin/true'
          archive_mode: on
          autovacuum_max_workers: 10
          autovacuum_vacuum_scale_factor: 0.05
          checkpoint_completion_target: 0.7
          default_statistics_target: 100
          effective_cache_size: 4GB
          effective_io_concurrency: 25
          hot_standby: on
          listen_addresses: '*'
          log_autovacuum_min_duration: 0
          log_checkpoints: on
          log_connections: off
          log_destination: syslog
          log_disconnections: off
          log_duration: off
          log_error_verbosity: default
          log_line_prefix: 'user=%u,db=%d,app=%a,client=%h '
          log_lock_waits: on
          log_min_duration_statement: 100
          log_temp_files: 0
          log_timezone: Europe/Moscow
          maintenance_work_mem: 2GB
          max_connections: 2000
```

```
          max_replication_slots: 10
          max_wal_senders: 10
          max_wal_size: 2GB
          min_wal_size: 1GB
          random_page_cost: 1.1
          shared_buffers: 1GB
          shared_preload_libraries: "pg_stat_statements"
          synchronous_commit: "off"
          syslog_facility: LOCAL6
          syslog_ident: postgres
          temp_buffers: 256MB
          timezone: Europe/Moscow
          wal_buffers: 16MB
          wal_keep_segments: 100
          wal_level: hot_standby
          work_mem: 41943kB
    initdb:
      - encoding: UTF8
      - data-checksums
    pg_hba:
      - local all all peer
      - host all all 127.0.0.1/32 trust
      - host all all ::1/128 trust
      - host all all 0.0.0.0/0 md5
      - host replication repl 0.0.0.0/0 md5
      - host replication repl ::1/128 md5
    users:
      repl:
        password: "repl"
        options:
          - replication
      repl_mon:
        password: "repl_mon"
        options:
          - replication
      postgres:
        password: "postgres"
        options:
          - superuser
      rewind:
        password: "rewind"
        options:
          - superuser
    post_bootstrap: /tmp/post_bootstrap.sh # post_bootstrap.sh dbname=postgres user=postgres
host=localhost port=5432
  postgresql:
      listen: '*:{{ postgresql_port }}'
      bin_dir: "{{ postgresql_bin_directory }}"
      connect_address: '{{ ansible_default_ipv4.address }}:{{ postgresql_port }}'
      data_dir: "{{ postgresql_data_directory }}"
      pgpass: "/pgsql/pgpass"
      pg_hba: "{{ postgresql_pg_hba }}"
      authentication:
        replication:
          username: repl
          password: "repl"
        superuser:
          username: postgres
          password: "postgres"
        rewind:
          username: rewind
          password: "rewind"
      parameters:
          unix_socket_directories: '/var/run/postgresql/'
  tags:
      clonefrom: False
      nofailover: "{{ patroni_nofailover | default(False) }}"
      noloadbalance: "{{ patroni_noloadbalance | default(False) }}"
      nosync: False
```

- Определяет конфигурацию patroni

Использование роли

Тестирование роли

> Для использования molecule необходимо наличие python модуля molecule>=3.6.1 и molecule-openstack>=0.3

Для тестирования ansible роли, используется molecule. Для запуска тестирования необходимо, находясь в корневой папке проекта ansible, вызвать следующую команду:

```
molecule converge -s postgresql
```

В результате, будет создан тестовый instance, запустится ansible role и тесты, которые проверят работоспособность кластера Postgresql.

После завершения тестов, сделает вызвать команду, которая удалит тестовый instance :

```
molecule destroy -s postgresql
```

Вызов роли

> Для использования Ansible roles необходимо наличие установленной Ansible

Для запуска Ansible role необходимо воспользоваться следующей командой :

```
ansible-playbook -i <inventory_file_path> playbook.yml --tags db --skip-tags always
```

Prometheus

- Общие сведения
- Ansible
  - Переменные роли
  - Использование роли
    - Тестирование роли
    - Вызов роли

Общие сведения

> официальный сайт

Prometheus - система мониторинга и сбора метрик от сервисов или о состоянии instance.

Так же, для сбора метрик, prometheus использует сервисы-помощники - exporters.

На текущий момент, к установке, доступен следующий список exporters:

- node_exporter
  - Порт: 9100
  - Собирает сведения об instance на которой развернут сервис.
- apache_exporter
  - Порт: 9117
  - Собирает сведения о состоянии веб сервиса apache.
- blackbox_exporter
  - Порт: 9115
  - Позволяет собирать пробы по принципу blackbox используя протоколы HTTP, HTTPS, DNS, TCP, ICMP и gRPC.
- consul_exporter
  - Порт: 9107
  - Собирает сведения об агентах consul.
- collectd_exporter
  - Порт: 9103
  - Собирает сведения о collectd.
- elasticsearch_exporter
  - Порт: 9114
  - Собирает сведения о состоянии elasticsearch.
- graphite_exporter
  - Порт: 9108
  - Собирает сведения о состоянии graphite.
- haproxy_exporter
  - Порт: 9101
  - Собирает сведения о состоянии haproxy.
- jmx_exporter
  - Порт: 12345
  - jmx.
- mysqld_exporter
  - Порт: 9104
  - Собирает сведения о состоянии mysql.
- nginx_exporter
  - Порт: 9113
  - Собирает сведения о nginx.
- postgres_exporter
  - Порт: 9187
  - Собирает сведения о состоянии postgresql.
- redis_exporter
  - Порт: 9121
  - Собирает сведения о состоянии redis.
- snmp_exporter
  - Порт: 9116
  - snmp.

Ansible

Ansible role

molecule

Переменные роли

tiers:

- Тип - список строк
- Значение по умолчанию - ["batch", "front"]
- При формировании файлов конфигурации, для каждого из элемента в списке будет создана отдельная группа и настроены адреса, по которым необходимо забирать метрики.

PROMETHEUS_GPG_KEYS:

- Тип - список строк
- Значение по умолчанию - ["http://mirror.i/centos/prometheus/gpgkey", "http://mirror.i/centos/prometheus/RPM-GPG-KEY-prometheus-rpm"]
- Используется при добавлении проверки, при установке компонентов prometheus.

EXPORTERS_PORTS:

Данный элемент используется при формировании конфигурационных файлом и не должен изменятся в сторону уменьшения значения.

- Тип - список элементов типа ключ-значени
- Значение по умолчанию -
  node_exporter: 9100
  apache_exporter: 9117
  blackbox_exporter: 9115
  consul_exporter: 9107
  collectd_exporter: 9103
  elasticsearch_exporter: 9114
  graphite_exporter: 9108
  haproxy_exporter: 9101
  jmx_exporter: 12345
  mysqld_exporter: 9104
  nginx_exporter: 9113
  postgres_exporter: 9187
  redis_exporter: 9121
  snmp_exporter: 9116
- При создании конфигурационных файлов, значения из списка будут взяты для сопоставления названия exporter, установленного на конкретном хосте с портом, который использует exporter.

COMPONENTS:

- Тип - список объектов
- Значение по умолчанию - отсутствует
- Используется для указания списка элементов prometheus, которые необходимо установить или удалить, так же используется для передачи параметров запуска компонента.

EX:

```
roles:
        - name: prometheus
          vars:
                - COMPONENTS:
                        - name: prometheus
                            install: true
                          opts: "<SOME_OPTIONS>"
                          - name: node_exporter
                            install: true
                          - name: apache_exporter
                            install: false
```

При вызове роли с конфигурацией описанной выше, будет установлен(и сконфигурирован) сервис prometheus, с параметрами указанными в строчку и сервис node_exporter, который автоматически попадет в список targets для сервиса prometheus. При этом, сервис apache_exporter установлен не будет, а в случае, если он уже присутствует на хосте, будет удален и убран из конфигурационного файла prometheus.

> При указании параметров в поле opts, необходимо указывать только те параметры, которые не являются параметрами по умолчанию, иначе они будут задвоены.

EX2:

```
roles:
        - name: prometheus
          vars:
                - COMPONENTS:
                        - name: node_exporter
                            install: true
```

В примере описано выше, будет вызвана роль, которая установит на хосте, только node_exporter.

> Т.к компонент prometheus будет использовать environment переменные на всех серверах, куда ansible сможет получить доступ, в поиске переменной COMPONENTS, то роль с компонентом prometheus должна быть вызвана в последнюю очередь, иначе, конфигурационные файлы с указанием targets хостов и портов exporters, используемых на них, сформированы не будут.

Использование роли

Тестирование роли

> Для использования molecule необходимо наличие python модуля molecule>=3.6.1 и molecule-openstack>=0.3

Для тестирования ansible роли, используется molecule. Для запуска тестирования необходимо, находясь в корневой папке проекта ansible, вызвать следующую команду:

```
molecule converge -s prometheus
```

В результате, будет создан тестовый instance, запустится ansible role и тесты, которые проверят работоспособность сервиса Prometheus и его exporters.

После завершения тестов, сделает вызвать команду, которая удалит тестовый instance :

```
molecule destroy -s prometheus
```

Вызов роли

> Для использования Ansible roles необходимо наличие установленной Ansible

Для запуска Ansible role необходимо воспользоваться следующей командой :

```
ansible-playbook -i <inventory_file_path> <playbook_path>.yaml
```

Redis

- Общие сведения
- Terraform
- Ansible
    - Переменные роли
    - Использование роли
        - Тестирование роли
        - Вызов роли

Общие сведения

официальный сайт

Redis - сервис, который позволяет хранить в памяти данные, которые получает от сервисов VKDoc.

Terraform

Использование terraform module требует наличия Terraform версии, не ниже `0.13,` `Openstack`

Для разворачивания инфраструктуры при помощи terraform необходимо воспользоваться terraform manifest.
Подробную инструкцию, об использовании terraform module, можно найти тут.

После того как terraform manifest будет складирован на локальную машину необходимо создать необходимые ресурсы, следующими командами :

```
terraform init #   , terraform
terraform apply --auto-approve #  terraform module       resources
```

флаг --auto-approve в команде terraform apply --auto-approve, необходим для того, чтобы автоматически подтвердить создание инфраструктуры

Ansible

Ansible role

molecule

Переменные роли

redis_data_dir:

- Тип - строка
- Значение по умолчанию - /var/lib/redis
- Определяет директорию, в которой будут хранится данные сервиса.

redis_log_dir:

- Тип - строка
- Значение по умолчанию -/var/log/redis
- Определяет директорию, в которой будут хранится логи сервиса.

redis_run_dir:

- Тип - строка
- Значение по умолчанию - /var/run/redis
- Определяет директорию в которой будут находится pid файлы сервиса.

redis_conf_dir:

- Тип - строка
- Значение по умолчанию - /etc
- Определяет директорию в которой сервис будет искать конфигурационные файлы.

redis_port:

- Тип - число
- Значение по умолчанию - 6379
- Определяет порт на котором будет работать сервис

redis_packages:

- Тип - список строк
- Значение по умолчанию - ["redis"]
- Определяет список пакетов, которые необходимо установить для работы сервиса.

maxmemory:

- Тип - строка
- Значение по умолчанию - 1024mb
- Определяет размер ограничения потребления памяти (RAM) сервисом.

maxclients:

- Тип - строка
- Значение по умолчанию - 100
- Определяет максимальное количество клиентов, которые могу быть подключены к сервису единовременно.

redis_cluster_replicas:

- Тип - число
- Значение по умолчанию - '{{ 0 if groups[mems] | length <= 1 else (groups[mems] | length | int -1) }}'
- Вычисляемое значение, определяет, какое количество сервисов, будет представлено в роли реплики. Количество реплик расчитывается из общего количества хостов -1.
  EX: Если общее количество хостов равно 2, то количество реплик будет равно 1.
  EX2: Если для сервиса Redis выделен всего 1 хост, то реплик у такого кластера не будет.

redis_node_list:

- Тип - список строк
- Значение по умолчанию - "{{ groups[mems] | map('extract', hostvars, ['ansible_default_ipv4', 'address']) | arraypermute([':']) | arraypermute([6379]) }}"
- Вычисляемое значение, определяет список адресов хостов, на которых разворачивается сервис, и их портов.

Если кластер Redis, представлен более чем 1 нодой, то количество сервисов на каждом хосте будет расчитываться по общему количество хостов.
EX: если в кластере участвует 3 хоста, то на каждом хосте будет установлено по 3 сервиса, которые будут работать на портах от 6379 до 6382.

Использование роли

Тестирование роли

Для использования molecule необходимо наличие python модуля molecule>=3.6.1 и molecule-openstack>=0.3

Для тестирования ansible роли, используется molecule. Для запуска тестирования необходимо, находясь в корневой папке проекта ansible, вызвать следующую команду:

```
molecule converge -s redis
```

В результате, будет создан тестовый instance, запустится ansible role и тесты, которые проверят работоспособность сервиса Redis.

После завершения тестов, сделает вызвать команду, которая удалит тестовый instance :

```
molecule destroy -s redis
```

Вызов роли

Для использования Ansible roles необходимо наличие установленной Ansible

Для запуска Ansible role необходимо воспользоваться следующей командой :

```
ansible-playbook -i <inventory_file_path> playbook.yml --tags mem --skip-tags always
```

Rsyslog

- Общие сведения
- Terraform
- Ansible
    - Переменные роли
    - Использование роли
        - Тестирование роли
        - Вызов роли

Общие сведения

официальный сайт

Сервис rsyslog позволяет принимать , складывать и хранить логи с различных хостов и различных приложений и сервисов.

Terraform

Использование terraform module требует наличия Terraform версии, не ниже `0.13,` `Openstack`

Для разворачивания инфраструктуры при помощи terraform необходимо воспользоваться terraform manifest.
Подробную инструкцию, об использовании terraform module, можно найти тут.

После того как terraform manifest будет складирован на локальную машину необходимо создать необходимые ресурсы, следующими командами :

```
terraform init #   , terraform
terraform apply --auto-approve #  terraform module       resources
```

флаг --auto-approve в команде terraform apply --auto-approve, необходим для того, чтобы автоматически подтвердить создание инфраструктуры

Ansible

Ansible role

molecule

Переменные роли

RSYSLOG_INCLUDE_CONFIG_PATH:

- Тип - строка
- Значение по умолчанию - /etc/rsyslog.d
- Определяет директорию, в которую будут будут складываться конфигурационные файлы.

MAX_MSG_SIZE:

- Тип - строка
- Значение по умолчанию - 8k

- Определяет максимальный размер сообщения принимаемое сервисом.

USE_UDP:

- Тип - булевое
- Значение по умолчанию - true
- Определяет, необходимость использования UDP порта для прослушивания запросов.

UDP_NAME:

- Тип - строка
- Значение по умолчанию - "imudp"
- Определяет название input module UDP порта в конфигурационном файле сервиса.

> Используется только при значении переменной USE_UDP = true.

UDP_PORT:

- Тип - строка
- Значение по умолчанию - "514"
- Определяет номер порта, на котором сервис будет слушать запросы UDP.

> Используется только при значении переменной USE_UDP = true.

UDP_ADDRESS:

- Тип - строка
- Значение по умолчанию - "0.0.0.0"
- Определяет адрес по которому сервис будет слушать запросы UDP.

> Используется только при значении переменной USE_UDP = true.

UDP_RATELIMIT:

- Тип - строка
- Значение по умолчанию - "5"
- Ограничение скорости получения запросов сервисом Rsyslog на порту UDP в секунду. Для отключения лимита необходимо установить значение: 0 (https://www.rsyslog.com/doc/v8-stable/configuration/modules/imuxsock.html#syssock-ratelimit-interval)

> Используется только при значении переменной USE_UDP = true.

USE_TCP:

- Тип - булевое
- Значение по умолчанию - true
- Определяет, необходимость использования TCP порта для прослушивания запросов.

TCP_NAME:

- Тип - строка
- Значение по умолчанию - "imtcp"
- Определяет название input module TCP порта в конфигурационном файле сервиса.

> Используется только при значении переменной USE_TCP = true.

TCP_PORT:

- Тип - строка
- Значение по умолчанию - "5514"
- Определяет номер порта, на котором сервис будет слушать запросы TCP.

> Используется только при значении переменной USE_TCP = true.

TCP_ADDRESS:

- Тип - строка
- Значение по умолчанию - "0.0.0.0"
- Определяет адрес по которому сервис будет слушать запросы TCP.

> Используется только при значении переменной USE_TCP = true.

TCP_RATELIMIT:

- Тип - строка
- Значение по умолчанию - "5"
- Ограничение скорости получения запросов сервисом Rsyslog на порту TCP в секунду.

> Используется только при значении переменной USE_TCP = true.

DEFAULT_CONFIG_ITEMS:

- Тип - список объектов
- Значение по умолчанию - [
  {name: "05-hids2",lines:[
  { pattern: 'auth.*,authpriv.* @hids2.i', path: /var/log/secure }
  ],
  {name: "50-default", lines:[
  {pattern: '*.info,mail.none,authpriv.none,cron.none', path: /var/log/messages},
  {pattern: 'authpriv.*', path: /var/log/secure},
  {pattern: 'cron.*', path: /var/log/cron},
  {pattern: 'mail.*', path: /var/log/maillog },
  {pattern: 'uucp,news.crit', path: /var/log/spooler }
  {pattern: local7.*, path: /var/log/boot.log }
  ]
  ]
- Определяет список правил, для сортировки логов и пути до их местонахождения.

Использование роли

Тестирование роли

> Для использования molecule необходимо наличие python модуля molecule>=3.6.1 и molecule-openstack>=0.3

Для тестирования ansible роли, используется molecule. Для запуска тестирования необходимо, находясь в корневой папке проекта ansible, вызвать следующую команду:

```
molecule converge -s rsyslog
```

В результате, будет создан тестовый instance, запустится ansible role и тесты, которые проверят работоспособность сервиса Rsyslog.

После завершения тестов, сделает вызвать команду, которая удалит тестовый instance :

```
molecule destroy -s rsyslog
```

Вызов роли

> Для использования Ansible roles необходимо наличие установленной Ansible

Для запуска Ansible role необходимо воспользоваться следующей командой :

```
ansible-playbook -i <inventory_file_path> <playbook_path>.yaml
```

Terraform

Ссылка на проект

Эта статья - разбор Terraform module для создания инфраструктуры.

В ней будут описаны используемые переменные, их взаимодействия с модулем, переопределение и использование модуля и примеры использования

- Переменные модуля
  - All variables
    - Переменная availability_zone
  - Instance variables
    - Переменная instance
    - Переменная vm_custom_volume
    - Переменная ansible_groups
    - Переменная existing_keypair
  - Network variables
    - Переменная is_need_new_network
    - Переменная network_name
    - Переменная subnets
    - Переменная is_need_load_balancer
    - Переменная dns_name
    - Переменная sg
    - Переменная existing_sg
- Примеры создания ресурсов с использованием terraform module.
  - Переопределение переменных модуля.
    - variables.tf
    - main.tf
    - Результат

Переменные модуля

Переменные в модуле разбиты по разным файлам, для облегчения поиска той или иной переменной.

Далее будут рассмотренные все переменные модуля и их значения.

All variables

Ссылка на файл

Переменная availability_zone

```
variable "availability_zone" {
  type  = string
  default = "OST1"
}
```

Переменная определяет в какой зоне доступности будут создаваться, практически все ресурсы, создаваемые модулем.

Для просмотра всех имеющихся зон доступности, необходимо воспользоваться следующей командой:

```
openstack availability zone list
```

Instance variables

Ссылка на файл

Переменная instance

```
variable "instance" {
  type  = object({
    name            = string
    count           = number
    image           = string
    flavor          = string
    boot_disk_size  = number
    subnet_type     = optional(string)
  })
  default = {
      name            = "instance"
      count           = 0
      image           = "CentOS-7.9.2009-v22.16.0"
      flavor          = "Basic-1-1"
      boot_disk_size  = 10
    }

  description = <<-EOF
  "
  name            - Instance batch name.
  count           - Count of created instances.
  image           - Image name which will be used from instance creating.
  flavor          - Name of flavor.
  boot_disk_size  - VM boot disk size.
  subnet_type     - Type of subnet where instance will be created. Type must be like a type in variable
"subnets". Default value - internal
  "
  EOF
}
```

Переменная представлена в виде объекта, которая отвечает за характеристики создаваемых instance, и имеет следующие значения:

- name
  - Тип - строка
  - Обязательный
  - Значение по умолчанию - instance
  - Отвечает за то, как будут называться созданные instances. EX: при названии "test" и count = 2, названия instance будут test-0 и test-1
- count
  - Тип - число
  - Обязательный
  - Значение по умолчанию - 0
  - Отвечает за кол-во создаваемых instance. Если значение по умолчанию не переопределено, то кол-во созданных instance будет равно 0, т.е новых instance создано не будет
- image
  - Тип - строка
  - Обязательный
  - Значение по умолчанию - CentOS-7.9.2009-v22.16.0
  - Отвечает за тип instance которые будут созданы, посмотреть все возможные типы (в случае с провайдером Openstack) можно командой:

    ```
    openstack image list
    ```

- flavor
  - Тип - строка
  - Обязательный
  - Значение по умолчанию - Basic-1-1

    - Отвечает за "размер" создаваемых instance, посмотреть все возможные варианты (в случае с провайдером Openstack) можно командой:

      ```
      openstack flavor list
      ```

- boot_disk_size
    - Тип - число
    - Обязательный
    - Значение по умолчанию - 10
    - Обозначает размер загрузочного диска instance.
- subnet_type
    - Тип строка
    - Не обязательный
    - Значение по умолчанию - нет
    - Используется для определения в какой подсети должны находится создаваемые instance. Доступна значения, которые указаны в свойстве type переменной subnets

Переменная vm_custom_volume

```
variable "vm_custom_volume" {
  type          = object({
    type              = string
    size              = number
    name              = string
    source_vol_id   = optional(string)
  })

  default       = null
  description   = <<-EOF
  "
  type          - Type of volume. EX: high-iops, ceph-ssd, ost1-nvme (openstack volume type list)
  size          - Size of custom volume in Gigabytes.
  name          - Name of volume.
  source_vol_id - Volume can use existing volume id, to clone it.
  "
  EOF
}
```

Переменная - необязательная. В случае, если для текущих instance не нужны дополнительные диски, переменную переопределять не нужно.

Переменная представлена в виде объекта, которая отвечает за характеристики вспомогательных дисков для instance и имеет следующие значения:

- type
    - Тип - строка
    - Обязательный
    - Значение по умолчанию - нет
    - Отвечает за тип диска, который будет присоединен к instance при создании.
      Все возможные типы (в случае с провайдером Openstack) можно увидеть командой:

      ```
      openstack volume type list
      ```

- size
    - Тип - число
    - Обязательный
    - Значение по умолчанию - нет
    - Отвечает за размер диска, в гигабайтах, который будет присоединен к instance.
- name
    - Тип - строка
    - Обязательный

    - Значение по умолчанию - нет
    - Отвечает за название дисков, которые будут присоединены к instance. EX: Если в переменный instance свойство count имеет значение 2, значение переменно name = "test" и переменная size = 10 и type = "high-iops", то к каждому из 2-х instance будет присоединен диск типа High-IOPS размером в 10 Gb и названием test-0 или test-1.
- source_vol_id
    - Тип - строка
    - Не обязательный
    - Значение по умолчанию - нет
    - В случае, если в облаке уже создан volume, который необходимо присоединить к создаваемым instance - то его ID необходимо указать в этом поле

Переменная `ansible_groups`

```
variable "ansible_groups" {
  type        = list(string)
  default     = [""]
  description = "Marked instances with ansible groups,  for using dynamic inventory"
}
```

Переменная представлена в виде списка строк, и определяет, какие теги будут добавлены к созданным instance, что может быть полезно, при работе с dynamic inventory в Ansible.

Переменная existing_keypair

```
variable "existing_keypair" {
  type        = string
  default     = ""
  description = "Existing keypair to set to instances"
}
```

В случае если ssh пара ключ-значение уже созданы и есть необходимость использовать ее для других instances, то название этой пары необходимо указать в переменной existing_keypair.

Network variables

> Ссылка на файл

Переменная is_need_new_network

```
variable "is_need_new_network" {
  type          = bool
  default       = false
  description   = "If true, will be created new network"
}
```

Переменная имеет булевое значение и отвечает за то, нужно ли создавать новые сети, и размещать в них instances.

> Если переменная имеет значение true, необходимо заполнить значения переменной subnets

Переменная network_name

```
variable "network_name" {
  type          = string
  default       = "MayDayCorpcloud"
}
```

Переменная, определяет название сети в которой будут созданы остальные подсети(определяется переменной subnets).
Если переменная is_need_new_network имеет значение false, тогда переменная будет обозначать название уже имеющейся сети.

Переменная subnets

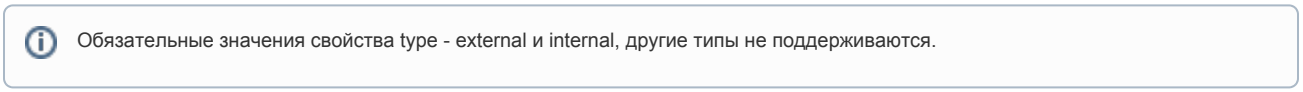
```
variable "subnets" {
  type  = list(object({
    name  = string
    type  = string
    cidr  = optional(string)
  }))

  default = [
    {
        name    = "MayDayPrivate"
        type    = "internal"
    },
    {
        name    = "MayDayCorpcloud"
        type    = "external"
    }
  ]

  description = "List of subnets"
}
```

Переменная имеет тип списка объектов, и определяет названия и типы сетей, в которых будут созданы instances.

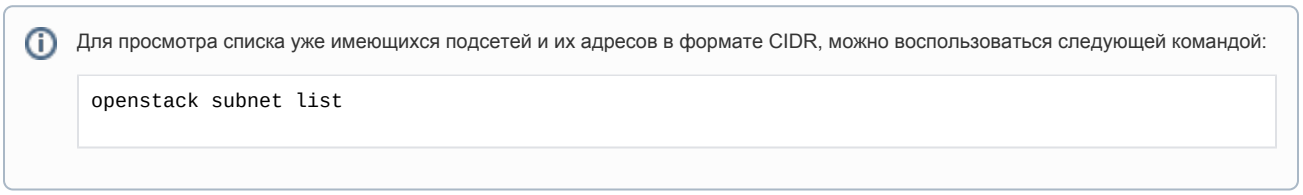
- name
  - Тип - строка
  - Обязательный
  - Значение. по умолчанию - MayDayPrivate и MayDayCorpcloud
  - Если переменная is_need_new_network имеет значение true, то названия сетей будут взяты из этой переменной. Если is_need_new_network - false, то имена подсетей будут использоваться для поиска уже существующих сетей. Посмотреть все существующие имена сетей можно командой

    ```
    openstack network list
    ```

- type

  - Тип - строка
  - Обязательный
  - Значение по умолчанию - internal и external
  - Используется для логического разделения типа сетей. Если свойство subnet_type не указано, то instance создаются в сети internal и имеют ассоциацию в сеть external.
- cidr

  - Тип - строка
  - Не обязательный
  - Переменная используется при создании новых сетей и позволяет задать точный диапазон адресов подсети, при этом, переменная is_need_new_network должна иметь значение true.

Переменная is_need_load_balancer

```
variable "is_need_load_balancer" {
        type = bool
        default = false
        description = "If true, load balancer will be created"
}
```

Переменная используется в сочетании с переменной sg, имеет булевый тип и отвечает за то, нужно ли создавать балансировщик нагрузки или нет.

Переменная dns_name

```
variable "dns_name" {
  type        = string
  default     = null

  description = "If is_need_load_balancer is true this variable will be taken to create dns name to load
balancer. If it is null or empty will take instance name."
}
```

Переменная имеет тип строки и дает возможность вручную задать DNS имя для балансировщика нагрузки. В случае, если переменная is_need_load_balancer имеет значение true, а переменная dns_name не имеет значения (поведение по умолчанию), то при создании балансировщика нагрузки, будет взято имя указанное в переменной instance свойства name.

Переменная sg

```
variable "sg" {
  type  = list(object({
    name            = string
    destination     = string
    protocol        = string
    port_range      = string

    load_balanced   = optional(bool)
    lb_protocol     = optional(string)
    lb_method       = optional(string)

    remote_ip_cidr  = optional(string)
    ip_type         = optional(string)
  }))
  default = [  ]
  description = <<-EOF
  "
  Instance security groups.
  name            - Name of security group.
  destination     - Ingress or Egress rule.
  protocol        - Protocol type:   tcp, udp, icmp, ah, dccp, egp, esp, gre, igmp, ipv6-encap, ipv6-frag, ipv6-
icmp, ipv6-nonxt, ipv6-opts, ipv6-route, ospf, pgm, rsvp, sctp, udplite, vrrp.
  port_range      - '-' separated port range. EX: 1-65535 or 22-22.
  load_balanced   - If true and is_need_load_balancer is true, then rule will be added to load balancer. !!
Will be getting a first number of port_range EX: 1-65535 will balanced port number 1!!
  lb_protocol     - Load balancer protocol. If load_balanced - true and lb_protocol is null, will take protocol
variable.
                    - Types: TCP, HTTP, HTTPS, PROXY, UDP (supported only in Octavia), PROXYV2 (Octavia minor
version >= 2.22) or SCTP (Octavia minor version >= 2.23).
  lb_method       - Load balancer method. If load_balanced - true and lb_method is null, will take ROUND_ROBIN.
                    - Types: ROUND_ROBIN, LEAST_CONNECTIONS, SOURCE_IP, or SOURCE_IP_PORT (supported only in
Octavia)
  remote_ip_cidr  - Remote ip cidr. Default value = 0.0.0.0/0.
  ip_type         - IPv4 or IPv6. Default - IPv4.
  "
  EOF
}
```

Переменная имеет тип списка объектов и позволяет работать с firewall на создаваемых instances, определяя порты и правила, которые будут созданы, при формирования instances.

> Указывая значение в переменной SG, правила будут создаваться(!) заново и иметь названия instance. Для использования уже имеющихся групп безопасности следует воспользоваться переменной existing_sg.

- name
  - Тип - строка
  - Обязательный
  - Значение по умолчанию - нет
  - Используется при формировании описания к правилу.
- destination
  - Тип - строка
  - Обязательный
  - Значение по умолчанию - нет
  - Определяет тип правила. Доступны два значения - ingress и egress.
- protocol
  - Тип - строка
  - Обязательный
  - Значение по умолчанию - нет
  - Определяет тип протокола для которого создается правило.
    Допустимые типы проторолов:
    - tcp
    - udp
    - icmp
    - ah
    - dccp
    - egp
    - esp
    - gre
    - igmp
    - ipv6-encap
    - ipv6-frag
    - ipv6-icmp
    - ipv6-nonxt
    - ipv6-opts
    - ipv6-route
    - ospf
    - pgm
    - rsvp
    - sctp
    - udplite
    - vrrp
- port_range

  > При балансировке определенных портов, возможно выбрать только протоколы TCP или UDP, при балансировке других протоколов, порты во внимание приняты не будут (!)

  - Тип - строка
  - Обязательный
  - Значение по умолчанию - нет
  - Диапазон портов, на которые распространяется создаваемое правило, записанное в формате `<PORT>-<PORT>` , где значения PORT могут быть как разными, так и одинаковыми.
    EX: запись формата 80-80 будет интерпретирована как единичный порт - 80. запись формата 1-65535 создаст правило для диапазона портов от 1 до 65535.
- load_balanced
  - Тип - булевое
  - Не Обязательный
  - Значение по умолчанию - нет
  - Если значение переменной is_need_load_balancer - true, то значение этого свойства будет определять, требуется ли создавать правило балансировки для нижнего значения портов.
    EX: если свойство load_balanced имеет значение true и свойство port_range имеет значение 1-6553, то при создании правила балансировки, будет взят только порт 1 (!)
- lb_protocol

> При балансировке определенных портов, возможно выбрать только протоколы TCP или UDP, при балансировке других протоколов, порты во внимание приняты не будут (!)

    - Тип - строка
    - Не Обязательный
    - Значение по умолчанию - нет
    - Определяет какой протокол использовать для балансировки нагрузки.
      Допустимые значения:
        - TCP
        - HTTP
        - HTTPS
        - PROXY
        - UDP
        - PROXYV2
        - SCTP
- lb_method
    - Тип - строка
    - Не Обязательный
    - Значение по умолчанию - ROUND_ROBIN
    - Определяет правило балансировки.
      Допустимые значения:
        - ROUND_ROBIN
        - LEAST_CONNECTIONS
        - SOURCE_IP
        - SOURCE_IP_PORT
- remote_ip_cidr
    - Тип - строка
    - Не Обязательный
    - Значение по умолчанию - 0.0.0.0/0
    - Определяет диапазон IP адресов, в формате CIDR, для которых создается правило балансировки.
- ip_type
    - Тип - строка
    - Не Обязательный
    - Значение по умолчанию - IPv4
    - Определяет версию IP адресов для создаваемого правила балансировки.

Переменная existing_sg

```
variable "existing_sg" {
    type    = list(string)
    default = [
        "default",
        "ssh+www",
        "ping"
    ]

    description = "List of sec groups which are already exists, and you want to add it to current instance"
}
```

Переменная имеет тип списка строк, и позволяет использовать уже существующие правила firewall, для создаваемых instances.
Для просмотра списка всех имеющихся групп безопасности, следует воспользоваться следующей командой:

```
openstack security group list
```

Примеры создания ресурсов с использованием terraform module.

В этой части статьи будет рассмотрен пример создания instance, sg, loadbalancer, вспомогательного диска типа ceph-ssd и использования уже существующих групп безопасности.

Переопределение переменных модуля.

variables.tf

Для создания группы instance, создайте 2 файла в одной папке - main.tf и variables.tf.

В файле variables.tf будут располагаться переменные, которые в дальнейшем, будут переопределены в модуле. Это могут быть любые переменные из приведенных выше, но мы рассмотрим только некоторые из них.

```
#    instance
variable "instance" {
  type  = object({
    name            = string
    count           = number
    image           = string
    flavor          = string
    boot_disk_size  = number
    subnet_name     = optional(string)
  })

  default = {
      name            = "test"
      count           = 2
      image           = "AlmaLinux-8-GenericCloud-8.6-20220513"
      flavor          = "Standard-4-16"
      boot_disk_size  = 10
      subnet_type     = "internal"
    }
}
#     instance
variable "vm_custom_volume" {
  type          = object({
    type                = string
    size                = number
    name                = string
    source_vol_id   = optional(string)
  })

  default       = {
        name               = "test-disk"
        type               = "ceph-ssd"
        size               = 50
    }
}
#
variable "sg" {
  type  = list(object({
    name            = string
    destination     = string
    protocol        = string
    port_range      = string

    load_balanced   = optional(bool)
    lb_protocol     = optional(string)
    lb_method       = optional(string)

    remote_ip_cidr  = optional(string)
    ip_type         = optional(string)
  }))
  default = [

      {
          name          = "http"
          destination   = "ingress"
          protocol      = "tcp"
          port_range    = "80-80"
          load_balanced = true
      },
      {
```

```
          name          = "https"
          destination   = "ingress"
          protocol      = "tcp"
          port_range    = "443-443"
      }
  ]
}
#
variable "existing_sg" {
    type    = list(string)
    default = [
        "default",
        "ssh+www",
        "ping"
    ]
```

При такой конфигурации, будет создано:

- 2 instance с названиями test-0 и test-1 с загрузочными дисками AlmaLinux-8 и размером в 10 Gb
- 2 вспомогательных диска типа ceph-ssd и размером 50 Gb, которые будут подключены к instance как НЕ загрузочные, в момент создания instance.
- 1 группа безопасности с названием test (свойство instance.name)
- 2 правила группы безопасности test:
    - http - использует протокол tcp на порту 80. Будет добавлено в правило балансировки, при создании балансировщика нагрузки.
    - https - использует протокол tcp на порту 443. При создании балансировщика нагрузки, правило балансировки создано не будет.

Так же, при создании instance будут переиспользованы 3 группы безопасности, которые называются default, ssh+www и ping.

main.tf

В файле main.tf будут описаны : провайдер, который необходимо использовать для работы модуля, переопределение переменных output (используются для получения информации о созданных ресурсах) и вызвать сам модуль.

```
#
terraform {
    required_providers {
        openstack = {
            source  = "terraform.local/local/openstack" #    Openstack
            version = "1.45.0" #
        }
    }
    required_version = ">=  0.13" #   terraform
    experiments      =  [module_variable_optional_attrs] # ,     optional
}

provider "openstack" {
  use_octavia = true #      Octavia
}

module "test" {
    source                = "../../modules/openstack" #
    instance              = var.instance #    instance,    variables.tf
    ansible_groups        = ["hrbox_test"] #    Ansible
    vm_custom_volume      = var.vm_custom_volume #    vm_custom_volume,    variables.tf
    sg                    = var.sg #    sg,    variables.tf
    is_need_load_balancer = true # ,
}

#   ssh ,     instance
output "ssh_public_key" {
  value = module.test.ssh_public_key
}
#  IP  instances
output "internal_ip" {
  value = module.test.instances_internal_ip
}
 #  IP  instances
output "external_ip" {
  value = module.test.instances_external_ip
}
```

При определении переменных типа output значение value должно принимать в значение название модуля.

EX: value = module.<MODULE_NAME>.instances_external_ip

Результат

Для создания необходимых ресурсов, в первую очередь необходимо, в папке с файлами конфигурации ввести команду *terraform init* , что позволит инициализировать необходимые для terraform файлы.

Далее представлен вывод команды *terraform plan* , для текущей конфигурации.

```
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated
with the following symbols:
  + create

Terraform will perform the following actions:

  # module.test.openstack_blockstorage_volume_v2.volume[0] will be created
  + resource "openstack_blockstorage_volume_v2" "volume" {
      + attachment        = (known after apply)
      + availability_zone = "OST1"
      + id                = (known after apply)
      + metadata          = (known after apply)
      + name              = "test-disk-0"
      + region            = (known after apply)
      + size              = 50
```

```
    + volume_type       = "ceph-ssd"
  }

# module.test.openstack_blockstorage_volume_v2.volume[1] will be created
+ resource "openstack_blockstorage_volume_v2" "volume" {
    + attachment        = (known after apply)
    + availability_zone = "OST1"
    + id                = (known after apply)
    + metadata          = (known after apply)
    + name              = "test-disk-1"
    + region            = (known after apply)
    + size              = 50
    + volume_type       = "ceph-ssd"
  }

# module.test.openstack_compute_floatingip_associate_v2.association[0] will be created
+ resource "openstack_compute_floatingip_associate_v2" "association" {
    + floating_ip = (known after apply)
    + id          = (known after apply)
    + instance_id = (known after apply)
    + region      = (known after apply)
  }

# module.test.openstack_compute_floatingip_associate_v2.association[1] will be created
+ resource "openstack_compute_floatingip_associate_v2" "association" {
    + floating_ip = (known after apply)
    + id          = (known after apply)
    + instance_id = (known after apply)
    + region      = (known after apply)
  }

# module.test.openstack_compute_instance_v2.instance[0] will be created
+ resource "openstack_compute_instance_v2" "instance" {
    + access_ip_v4        = (known after apply)
    + access_ip_v6        = (known after apply)
    + all_metadata        = (known after apply)
    + all_tags            = (known after apply)
    + availability_zone   = "OST1"
    + flavor_id           = (known after apply)
    + flavor_name         = "Standard-4-16"
    + force_delete        = false
    + id                  = (known after apply)
    + image_id            = "938dc1cf-ec86-47b5-9994-4462ec671ac0"
    + image_name          = (known after apply)
    + key_pair            = "test-ssh-keypair"
    + metadata            = (known after apply)
    + name                = "test-0"
    + power_state         = "active"
    + region              = (known after apply)
    + security_groups     = [
        + "default",
        + "ping",
        + "ssh+www",
        + "test",
      ]
    + stop_before_destroy = false

    + block_device {
        + boot_index            = 0
        + delete_on_termination = true
        + destination_type      = "volume"
        + source_type           = "image"
        + uuid                  = "938dc1cf-ec86-47b5-9994-4462ec671ac0"
        + volume_size           = 10
      }
    + block_device {
        + boot_index            = -1
        + delete_on_termination = true
        + destination_type      = "volume"
        + source_type           = "volume"
        + uuid                  = (known after apply)
```

```
      }

    + network {
        + access_network = false
        + fixed_ip_v4    = (known after apply)
        + fixed_ip_v6    = (known after apply)
        + floating_ip    = (known after apply)
        + mac            = (known after apply)
        + name           = "MayDayPrivate"
        + port           = (known after apply)
        + uuid           = (known after apply)
      }
  }

# module.test.openstack_compute_instance_v2.instance[1] will be created
+ resource "openstack_compute_instance_v2" "instance" {
    + access_ip_v4        = (known after apply)
    + access_ip_v6        = (known after apply)
    + all_metadata        = (known after apply)
    + all_tags            = (known after apply)
    + availability_zone   = "OST1"
    + flavor_id           = (known after apply)
    + flavor_name         = "Standard-4-16"
    + force_delete        = false
    + id                  = (known after apply)
    + image_id            = "938dc1cf-ec86-47b5-9994-4462ec671ac0"
    + image_name          = (known after apply)
    + key_pair            = "test-ssh-keypair"
    + metadata            = (known after apply)
    + name                = "test-1"
    + power_state         = "active"
    + region              = (known after apply)
    + security_groups     = [
        + "default",
        + "ping",
        + "ssh+www",
        + "test",
      ]
    + stop_before_destroy = false

    + block_device {
        + boot_index            = 0
        + delete_on_termination = true
        + destination_type      = "volume"
        + source_type           = "image"
        + uuid                  = "938dc1cf-ec86-47b5-9994-4462ec671ac0"
        + volume_size           = 10
      }
    + block_device {
        + boot_index            = -1
        + delete_on_termination = true
        + destination_type      = "volume"
        + source_type           = "volume"
        + uuid                  = (known after apply)
      }

    + network {
        + access_network = false
        + fixed_ip_v4    = (known after apply)
        + fixed_ip_v6    = (known after apply)
        + floating_ip    = (known after apply)
        + mac            = (known after apply)
        + name           = "MayDayPrivate"
        + port           = (known after apply)
        + uuid           = (known after apply)
      }
  }

# module.test.openstack_compute_keypair_v2.ssh_keypair will be created
+ resource "openstack_compute_keypair_v2" "ssh_keypair" {
    + fingerprint = (known after apply)
```

```
    + id          = (known after apply)
    + name        = "test-ssh-keypair"
    + private_key = (known after apply)
    + public_key  = (known after apply)
    + region      = (known after apply)
  }

# module.test.openstack_lb_listener_v2.lb_listener["http"] will be created
+ resource "openstack_lb_listener_v2" "lb_listener" {
    + admin_state_up         = true
    + connection_limit       = (known after apply)
    + default_pool_id        = (known after apply)
    + id                     = (known after apply)
    + loadbalancer_id        = (known after apply)
    + name                   = "test-http"
    + protocol               = "TCP"
    + protocol_port          = 80
    + region                 = (known after apply)
    + tenant_id              = (known after apply)
    + timeout_client_data    = (known after apply)
    + timeout_member_connect = (known after apply)
    + timeout_member_data    = (known after apply)
    + timeout_tcp_inspect    = (known after apply)
  }

# module.test.openstack_lb_loadbalancer_v2.main[0] will be created
+ resource "openstack_lb_loadbalancer_v2" "main" {
    + admin_state_up        = true
    + flavor_id             = (known after apply)
    + id                    = (known after apply)
    + loadbalancer_provider = "amphora"
    + name                  = "test-loadbalancer"
    + region                = (known after apply)
    + security_group_ids    = (known after apply)
    + tenant_id             = (known after apply)
    + vip_address           = (known after apply)
    + vip_network_id        = (known after apply)
    + vip_port_id           = (known after apply)
    + vip_subnet_id         = (known after apply)
  }

# module.test.openstack_lb_members_v2.members["http"] will be created
+ resource "openstack_lb_members_v2" "members" {
    + id      = (known after apply)
    + pool_id = (known after apply)
    + region  = (known after apply)

    + member {
        + address        = (known after apply)
        + admin_state_up = true
        + id             = (known after apply)
        + protocol_port  = 80
        + weight         = 1
      }
    + member {
        + address        = (known after apply)
        + admin_state_up = true
        + id             = (known after apply)
        + protocol_port  = 80
        + weight         = 1
      }
  }

# module.test.openstack_lb_pool_v2.main["http"] will be created
+ resource "openstack_lb_pool_v2" "main" {
    + admin_state_up  = true
    + id              = (known after apply)
    + lb_method       = "ROUND_ROBIN"
    + loadbalancer_id = (known after apply)
    + name            = "test-http"
    + protocol        = "TCP"
```

```
    + region          = (known after apply)
    + tenant_id       = (known after apply)

    + persistence {
        + cookie_name = (known after apply)
        + type        = (known after apply)
      }
  }

# module.test.openstack_networking_floatingip_associate_v2.lb_association[0] will be created
+ resource "openstack_networking_floatingip_associate_v2" "lb_association" {
    + fixed_ip    = (known after apply)
    + floating_ip = (known after apply)
    + id          = (known after apply)
    + port_id     = (known after apply)
    + region      = (known after apply)
  }

# module.test.openstack_networking_floatingip_v2.float[0] will be created
+ resource "openstack_networking_floatingip_v2" "float" {
    + address    = (known after apply)
    + all_tags   = (known after apply)
    + dns_domain = (known after apply)
    + dns_name   = (known after apply)
    + fixed_ip   = (known after apply)
    + id         = (known after apply)
    + pool       = "MayDayCorpcloud"
    + port_id    = (known after apply)
    + region     = (known after apply)
    + subnet_id  = (known after apply)
    + tenant_id  = (known after apply)
  }

# module.test.openstack_networking_floatingip_v2.float[1] will be created
+ resource "openstack_networking_floatingip_v2" "float" {
    + address    = (known after apply)
    + all_tags   = (known after apply)
    + dns_domain = (known after apply)
    + dns_name   = (known after apply)
    + fixed_ip   = (known after apply)
    + id         = (known after apply)
    + pool       = "MayDayCorpcloud"
    + port_id    = (known after apply)
    + region     = (known after apply)
    + subnet_id  = (known after apply)
    + tenant_id  = (known after apply)
  }

# module.test.openstack_networking_floatingip_v2.lb_float[0] will be created
+ resource "openstack_networking_floatingip_v2" "lb_float" {
    + address    = (known after apply)
    + all_tags   = (known after apply)
    + dns_domain = (known after apply)
    + dns_name   = (known after apply)
    + fixed_ip   = (known after apply)
    + id         = (known after apply)
    + pool       = "MayDayCorpcloud"
    + port_id    = (known after apply)
    + region     = (known after apply)
    + subnet_id  = (known after apply)
    + tenant_id  = (known after apply)
  }

# module.test.openstack_networking_port_v2.port[0] will be created
+ resource "openstack_networking_port_v2" "port" {
    + admin_state_up         = (known after apply)
    + all_fixed_ips          = (known after apply)
    + all_security_group_ids = (known after apply)
    + all_tags               = (known after apply)
    + device_id              = (known after apply)
    + device_owner           = "Octavia"
```

```
    + dns_assignment         = (known after apply)
    + dns_name               = "test"
    + id                     = (known after apply)
    + mac_address            = (known after apply)
    + name                   = "test-loadbalancer-port"
    + network_id             = "5527a7b4-cb5d-4042-a0e0-b2cfd52f12f3"
    + port_security_enabled  = (known after apply)
    + qos_policy_id          = (known after apply)
    + region                 = (known after apply)
    + tenant_id              = (known after apply)

    + binding {
        + host_id     = (known after apply)
        + profile     = (known after apply)
        + vif_details = (known after apply)
        + vif_type    = (known after apply)
        + vnic_type   = (known after apply)
      }
  }

# module.test.openstack_networking_secgroup_rule_v2.sg_rule[0] will be created
+ resource "openstack_networking_secgroup_rule_v2" "sg_rule" {
    + description       = "http"
    + direction         = "ingress"
    + ethertype         = "IPv4"
    + id                = (known after apply)
    + port_range_max    = 80
    + port_range_min    = 80
    + protocol          = "tcp"
    + region            = (known after apply)
    + remote_group_id   = (known after apply)
    + remote_ip_prefix  = "0.0.0.0/0"
    + security_group_id = (known after apply)
    + tenant_id         = (known after apply)
  }

# module.test.openstack_networking_secgroup_rule_v2.sg_rule[1] will be created
+ resource "openstack_networking_secgroup_rule_v2" "sg_rule" {
    + description       = "https"
    + direction         = "ingress"
    + ethertype         = "IPv4"
    + id                = (known after apply)
    + port_range_max    = 443
    + port_range_min    = 443
    + protocol          = "tcp"
    + region            = (known after apply)
    + remote_group_id   = (known after apply)
    + remote_ip_prefix  = "0.0.0.0/0"
    + security_group_id = (known after apply)
    + tenant_id         = (known after apply)
  }

# module.test.openstack_networking_secgroup_v2.instance_sg[0] will be created
+ resource "openstack_networking_secgroup_v2" "instance_sg" {
    + all_tags             = (known after apply)
    + delete_default_rules = false
    + description          = (known after apply)
    + id                   = (known after apply)
    + name                 = "test"
    + region               = (known after apply)
    + tenant_id            = (known after apply)
  }

# module.test.openstack_networking_secgroup_v2.instance_sg[1] will be created
+ resource "openstack_networking_secgroup_v2" "instance_sg" {
    + all_tags             = (known after apply)
    + delete_default_rules = false
    + description          = (known after apply)
    + id                   = (known after apply)
    + name                 = "test"
    + region               = (known after apply)
```

```
      + tenant_id            = (known after apply)
    }

Plan: 20 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + external_ip    = [
      + (known after apply),
      + (known after apply),
    ]
  + internal_ip    = [
      + (known after apply),
      + (known after apply),
    ]
  + ssh_public_key = (known after apply)
```

Стек ELK

- Общие сведения
- Terraform
- Ansible
    - Переменные роли
    - Использование роли
        - Тестирование роли
        - Вызов роли

Общие сведения

официальный сайт

Стек ELK используется для сбора и визуализации логов приложений и сервисов.

В стандартный стек ELK входят следующие сервисы:

- Elasticsearch
- Logstash
- Kibana

В конфигурации on-premise используются сервисы:

- Elasticsearch
- Kibana
- APM

Terraform

Использование terraform module требует наличия Terraform версии, не ниже `0.13,` `Openstack`

Для разворачивания инфраструктуры при помощи terraform необходимо воспользоваться terraform manifest.
Подробную инструкцию, об использовании terraform module, можно найти тут.

После того как terraform manifest будет складирован на локальную машину необходимо создать необходимые ресурсы, следующими командами :

```
terraform init #   , terraform
terraform apply --auto-approve #  terraform module       resources
```

флаг --auto-approve в команде terraform apply --auto-approve, необходим для того, чтобы автоматически подтвердить создание инфраструктуры

Ansible

Ansible role

molecule

Переменные роли

ELK_VERSION:

- Тип - строка

- Значение по умолчанию - 7.17.1
- Определяет версию каждого компонента стека.

ELK_REPO:

- Тип - строка
- Значение по умолчанию - 'http://mirror.i.mail.ru/centos/elasticsearch/7.x/'
- Задает URL до ресурса, на котором расположены rpm пакеты elastic стека.

ELK_IFACE:

- Тип - строка
- Значение по умолчанию - 'eth0'
- Определяет сетевой интерфейс на котором слушают сервисы. Необходим, для корректного определения IP адресов хостов.

ELASTIC_DATA_DIR:

- Тип - строка
- Значение по умолчанию - '/data/elasticsearch'
- Определяет директорию хранения данных сервиса elasticsearch.

ELASTIC_LOGS_DIR:

- Тип - строка
- Значение по умолчанию - /var/logs/elasticsearch
- Определяет директорию, для хранения логов сервиса elasticsearch.

ELASTIC_CONFIG_PATH:

- Тип - строка
- Значение по умолчанию - /etc/elasticsearch
- Определяет директорию, в которой располагаются конфигурационные файлы, для сервиса elasticsearch.

Использование роли

Тестирование роли

Для использования molecule необходимо наличие python модуля molecule>=3.6.1 и molecule-openstack>=0.3

Для тестирования ansible роли, используется molecule. Для запуска тестирования необходимо, находясь в корневой папке проекта ansible, вызвать следующую команду:

```
molecule converge -s ELK
```

В результате, будет создан тестовый instance, запустится ansible role и тесты, которые проверят работоспособность стека ELK.

После завершения тестов, сделает вызвать команду, которая удалит тестовый instance :

```
molecule destroy -s ELK
```

Вызов роли

Для использования Ansible roles необходимо наличие установленной Ansible

Для запуска Ansible role необходимо воспользоваться следующей командой :

```
ansible-playbook -i <inventory_file_path> playbook.yml --tags elk --skip-tags always
```