Р7-ОФИС

Инструкцию по установке CS в Middle архитектуре при помощи инструментов ansible и terraform

Содержание

1. Архитектура

В данной инструкции рассмотрим скрипты автоматизации деплоя и организации необходимых тестовых контуров в автоматическом режиме программного продукта Р7 Сервер Базовый в архитектуре Middle.

Ниже представлена схема архитектуры Middle

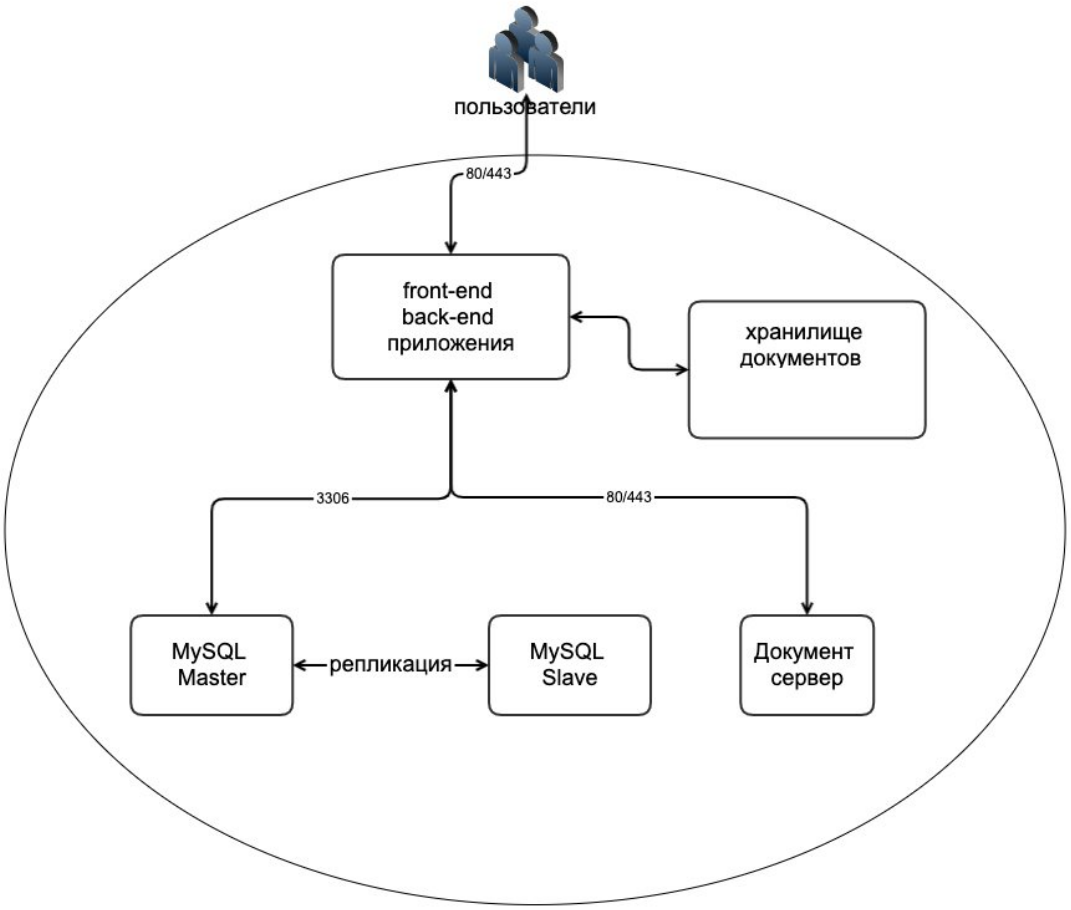

схема middle архитекруры

Преимущества данной архитектуры заключаются в повышении отказоустойчивости системы в целом и снижении нагрузки на сервер приложения путём выноса функциональных модулей.

1.1 Front-end / back-end приложения

Как в случае с Microsoft Windows, так и в случае с Linux версией продукта, front-end и back-end приложения размещается на одном сервере. Нагрузка на сервер приложений снижается за счёт размещения функциональных модулей на отдельных серверах.

1.2 Система Управления Базами Данных.

База данных является неотъемлемой частью Продукта, обеспечивает хранение и управление следующими данными (укрупнённо):

- информация о пользователях системы;
- содержится мета информация для документов и писем;
- пользовательские данные по рабочим модулям Продукта.

Для устранения факторов, влияющих на деградацию производительности Программного комплекса в целом, в архитектурном решении Middle, система управления базами данных устанавливается на отдельные сервера для исключения воздействия сторонних систем, которые могут привести, в том числе, и к увеличению времени ожидания диска, что является одним из критических показателей для информационных систем.

1.3 Документ сервер

Система документ сервера в данном варианте инсталляции предусматривает размещение на отдельных серверах как в версии Docker контейнера так и с применением обычной установки.

Функционирование документ сервера возможно как на базе операционных систем типа Microsoft Windows так и Linux подобных системах.

2. Создание инфраструктуры

2.1 Terraform

В качестве сборки инфраструктуры предлагаем рассмотреть манифест terrafrom и playbooks ansible для openstack.

Ниже представлен код для создания ВМ в terraform (перемененные правильно выносить в отдельный файл, но можно записать и в основной файл с кодом):

```
# описание поставщиков, которые задействованы при выполнение
terraform {
required_version = ">= 0.14.0"
  required_providers {
    openstack = {
      source  = "terraform-provider-openstack/openstack"
      version = "~> 1.47.0"
    }
     selectel = {
      source  = "selectel/selectel"
      version = "~> 3.8.4"
   }
  }
}

# В данном блоке описываем подключение к провайдеру openstack
provider "openstack" {
  auth_url    = "https://api.selvpc.ru/identity/v3"
  domain_name = var.account_selectel
  tenant_id   = var.project_id
  user_name   = var.user_openstack
  password    = var.pwd_openstack
  region      = var.region
}

# В данном блоке добавляем token для доступа в selectel
provider "selectel" {
  token = var.token_selectel
}

# Данный ресурс создаёт открытый ключ на площадке
resource "openstack_compute_keypair_v2" "devops_key" {
  name       = var.key_name
  region     = var.region
  public_key = file(var.public_key)
}

# На этом этапе мы создаём диск с нужным нам образом ОС, задавая размер
# зону, тип и имя его
```

```
resource "openstack_blockstorage_volume_v3" "volume_vm" {
  count             = "${length(var.name_vm)}"
  name              = var.name_vm[count.index]
  size              = var.size_disk[count.index]
  image_id          = var.image_id
  availability_zone = var.avail_zone
  volume_type       = var.volume_type
}

# Финальный этап, создание ВМ
resource "openstack_compute_instance_v2" "devops_vm" {
# Количество ресурсов, которое необходимо создать
  count             = "${length(var.name_vm)}"
# ИМЯ ВМ
  name              = var.name_vm[count.index]
# Параметр отвечает за CPU-RAM ВМ
  flavor_name       = var.flavor[count.index]
# id ключа, который мы создали
  key_pair          = openstack_compute_keypair_v2.devops_key.id
  security_groups   = ["default"]
# Зона, в которой создаём ВМ
  availability_zone = var.avail_zone

# Описываем диск, на основе которого ВМ будет создана
  block_device {
    uuid                  =
openstack_blockstorage_volume_v3.volume_vm[count.index].id
    source_type           = "volume"
    boot_index            = 0
    destination_type      = "volume"
    delete_on_termination = true
  }

 # Сетевые интерфейсы, которые будут добавлены на ресурс
  network {
    name = var.network[0]
  }


  network {
    name = var.network[1]
  }


}
```

Перед запуском создания инфраструктуры с помощью terraform необходимо выполнить команду для инициализации поставщиков:

```
terraform init
```

После этого, если всё указано корректно, выполним команду для проверки корректности кода и того, что будет создано:

```
terraform plan
```

Далее запускаем команду для создание ресурсов:

```
terraform apply
```

Если не используете terraform, то можно создать инфраструктуру ещё несколькими способами:

2.2 Openstack CLI

Необходимо настроить доступ перед запуском, у провайдера должна быть инструкция для этого

```
# Добавляем открытый ключ
openstack keypair create <имя ключа> --public-key <путь до открытого ключа>

# Создаём диск
openstack volume create \
    --image <id образа> \
    --size <размер диска> \
    --type <тип диска> \
    --availability-zone <зона, в которой он будет создан> \
   <имя диска>

# Создаём сервер
openstack server create \
    --volume <id созданного диска> \
    --flavor <id flavor> \
    --availability-zone <зона, в которой он будет создан> \
    --key-name <имя созданного ранее ключа> \
    --nic net-id=<id сети> \
     <имя виртуальной машины>
```

Для получения нужного количества ресурсов – сделать дополнительно 3 итерации создания диска и ВМ с верными характеристиками.

2.3 Ansible playbooks

Примечание: необходимо настроить доступ в Openstacl CLI перед запуском, у провайдера должна быть инструкция для этого.

```
- name: Create disk
  os_volume:
    state: present
    bootable: yes
    availability_zone: "{{ availability_zone }}"
    size: "{{ disk_size }}"
    image: "{{ UUID_image }}"
    volume_type: "{{ volume_type }}"
    display_name: "{{ disk_name }}"
  register: volume_info

- name: Create a new instance
  os_server:
    name: "{{ server_name }}"
    boot_volume: "{{ volume_info.volume.name }}"
    key_name: "{{ public_key }}"
    timeout: 200
    availability_zone: "{{ availability_zone }}"
    auto_ip: no
    network: nat
    flavor: "{{ flavor }}"
  register: vm_info
```

Предварительно создать ключ. Переменные можно вынести отдельно или задавать при запуске. Пример команды:

```
ansible-playbook create_instans.yaml -e UUID_image=<id образа> -e
public_key=<имя созданного ключа> -e flavor=<id flavor> -e volume_type=<тип
диска> -e server_name=<имя сервера> -e disk_name=<имя диска> -e
disk_size=<размер диска>
```

3. Автоматизированные скрипты установки Ansible

Подготовив инфраструктура для установки продукта, перейдём к playbooks и ролям.

Playbok deploy_middle.yaml содержит запуск следующих ролей:

```
---
# Устанавливает MySQL на машинах для СУБД
- hosts: mysql
  gather_facts: True
  become: True
  roles:
    - deploy_mysql

# Настраивает мастер ноду СУБД
- hosts: master_db
  gather_facts: True
  become: True
  roles:
    - mysql_master


# Настраивает slave ноду
- hosts: slave_db
  gather_facts: True
  become: True
  tasks:
    - name: slave role
      include_role:
        name: mysql_slave
      when: mysql_cluster == "true"
# Запуск роли для инсталляции сервера документов
- hosts: ds
  gather_facts: True
  become: True
  roles:
    - deploy_DS
# Запуск роли для инсталляции Сервера Совместной Работы
- hosts: cs
  gather_facts: True
  become: True
  roles:
    - deploy_CS
```

В перемененные для всех ВМ выносятся данные:

```
---
# for all vm
ansible_port: 2235
ansible_user: "rhel"

# for cs and nginx
cs_cluster: "false"
#domain_name: cs_cluster.ru
vars_ssl: false

# dor ds and cs
jwt_secret: "JWT токен доступа для CS и DS"
jwt_header: AuthorizationJwt

# for mysql and cs
mysql_pass: "пароль для пользователя"
mysql_user: "root"
# имя пользователя можно изменить, как для обычного пользователя,
# так и для реплики
mysql_user_repl: "slave_repl"
mysql_pass_repl: "пароль для пользователя репликации"
# задать false, если будет только одна ВМ с СУБД
mysql_cluster: "true"
```

Их можно задать двумя способами:

- изменить в файле с переменными
- задать через ключ --extra-vars

Далее в инвентори необходимо указать верные данные для подключения ansible к hosts:

```
[middle:children]
cs
ds
mysql

[cs:children]
cs_master

[cs_master]
cs_1       ansible_host=192.168.0.09

[ds]
ds_node          ansible_host=192.168.0.10

[mysql:children]
master_db
slave_db

[master_db]
master_node      ansible_host=192.168.0.11

[slave_db]
slave_node       ansible_host=192.168.0.12
```

А в ansible.cfg () указать путь к закрытому ключу и к файлу с паролем для расшифровки чувствительных переменных (последнее, если шифровали переменные):

```
[defaults]
...
# пути указаны, как пример
vault_password_file = ~/.ansible_secret
private_key_file = ~/devops_key
...
```

Остался последний пункт, запуск плейбука для настройки и инсталляции. Запуск происходит обычной командой:

```
ansible-playbook deploy_middle.yaml
```

Либо с заданием переменных, если не хотите изменять конфигурационный файл:

```
ansible-playbook --vault-password-file /home/r7_user/.ansible_secret --private-
key ~/.ssh/SSHKEY deploy_middle.yaml -e mysql_user=r7_user -e
mysql_pass=12345QWEasdZXC! -e
```

Далее остаётся дождаться результата и зарегистрироваться на портале.