

Если вы видите что-то необычное, просто сообщите мне.

Пример ansible для установки и настройки docker swarm

Описание

Что необходимо на входе

4 хоста с предустановленной debian ОС на борту. В моем случае это 4 виртуальные машины созданные методом копирования пустой установленной системы Debian ОС.

Что получим на выходе

Кластер docker-swarm с 4мя нодами: 1 менеджер и 3 воркера Кластер glusterfs на всех 4х нодах. Файл размещенные по пути /mnt будут синхронизироваться на всех хостах. Это позволит запускать контейнеры на любой ноде зная, что данные будут доступны для любого контейнера на любой ноде.

Необходимые ansible параметры для настройки ХОСТОВ

Подготавливаем inventory.ini

```
[manager]
manager  ansible_host=192.168.1.46 ansible_user=USERNAME
[nodes]
node1 ansible_host=192.168.1.61 ansible_user=USERNAME
node2 ansible_host=192.168.1.67 ansible_user=USERNAME
node3 ansible_host=192.168.1.96 ansible_user=USERNAME
```

В файле мы разделили хосты на 2 группы:

- manager - хосты для управления кластером
- nodes - хосты для работы кластера

Необходимо указать

- имена хостов: `manager`, `node1`, `node2`, `node3` указаны для примера, эти имена будут использоваться для того, чтобы задать имена хостам.
- `ansible_host` - ip адрес хоста
- `ansible_user` - пользователь под которым будет происходить настройка сервера.

“ в случае использования Debian часто приходится отдельно добавлять пользователя USERNAME в группу sudo. Для этого нужно установить sudo, и добавить пользователя в группу sudo командой под супер пользователем.

```
su // ввести root пароль
/usr/sbin/useradd -aG sudo USERNAME
```

Playbook для настройки сервера

Данный playbook выполняется для всех хостов в файле inventory.ini

```
#play-hostconfig.yaml
- name: Prepear hosts
  hosts: all
  become: true
  tasks:
    - name: Изменение имени хоста
      ansible.builtin.hostname:
        name: "{{ inventory_hostname }}"

    # Шаг может быть пропущен если вы используете внешний dns
    - name: Создаем dns записи для нашего кластера
      lineinfile:
        dest: /etc/hosts
        regexp: '.*{{ item }}$'
        line: '{{ hostvars[item].ansible_default_ipv4.address }} {{item}}'
        state: present
      with_items: '{{ groups["all"] }}'

- name: Установка пакетов для поддержки HTTPS в apt
  apt:
    name:
      - apt-transport-https
      - ca-certificates
      - curl
      - gnupg2
      - software-properties-common
    state: present
```

Установка glusterfs

Приведенный пример ниже, устанавливает на 4 хоста glusterfs - создает папку и сервис который синхронизирует её между серверами, а так же устанавливает docker.

```
#play-glusterfs.yaml
---
- name: Устанавливаем необходимые компоненты
  hosts: all
  become: true
  tasks:
    - name: Обновляем кэш
      apt:
        update_cache: true

    - name: Устанавливаем GlusterFS сервер
      apt:
        name: glusterfs-server
        state: present

    - name: Запускаем glusterd сервис
      systemd:
        name: glusterd
        state: started
        enabled: true

    - name: Создаем /gluster/volumes папку
      file:
        path: /gluster/volumes
        state: directory
        mode: '0755'
        owner: root
        group: root

- name: Применяем GlusterFS конфигурацию для manager
  hosts: manager
```

```
become: true
tasks:

- name: Проверяем доступность нод
  command: "gluster peer probe {{ item }}"
  ignore_errors: true
  with_items: '{{ groups["nodes"] }}'

# Требуется доступности хостов друг другу во всех направлениях
- name: Create gluster volume staging-gfs
  command: >
    gluster volume create staging-gfs replica 4
    manager:/gluster/volumes
    node1:/gluster/volumes
    node2:/gluster/volumes
    node3:/gluster/volumes
    force
  register: volume_create_result
  changed_when: "'Volume successfully created' in volume_create_result.stdout"
  ignore_errors: true

- name: Создаем раздел staging-gfs для glusterfs
  command: gluster volume start staging-gfs
  ignore_errors: true

- name: Создаем автоматическое монтирование раздела staging-gfs на всех хостах
  hosts: all
  serial: 1
  become: true
  tasks:

- name: Обновляем fstab
  lineinfile:
    dest: /etc/fstab
    regexp: '.*staging-gfs.*'
    #line: localhost:/staging-gfs /mnt glusterfs defaults,_netdev,backupvolfile-
server=localhost,x-systemd.requires=network-online.target 0 0
    line: localhost:/staging-gfs /mnt glusterfs defaults,_netdev,noauto,x-
```

```
systemd.automount 0 0
    state: absent

- name: Создаем задачу при запуске системы "@reboot mount.glusterfs localhost:/staging-gfs
/mnt"
  ansible.builtin.cron:
    name: "mount for reboot"
    special_time: reboot
    job: "sleep 20 && /usr/sbin/mount.glusterfs localhost:/staging-gfs /mnt >
/root/mount.log 2>&1"
    state: present

- name: Перезагружаем сервер после завершения конфигурации
  ansible.builtin.reboot:
```

Заведение серверов в dockekr swarm.

```
#play-docker.yaml
---
- name: Установка docker
  hosts: all
  become: true
  tasks:
    - name: Добавление GPG-ключа Docker
      apt_key:
        url: https://download.docker.com/linux/debian/gpg
        state: present

    - name: Добавление репозитория Docker
      apt_repository:
        repo: deb https://download.docker.com/linux/debian buster stable
        state: present

- name: Установка Docker Engine
```

```
apt:  
  name: docker-ce  
  state: present
```

```
- name: Добавление текущего пользователя в группу docker
```

```
user:  
  name: user  
  append: yes  
  groups: docker
```

```
- name: Меняем владельца на docker для /mnt папки
```

```
file:  
  path: /mnt  
  owner: root  
  group: docker  
  recurse: yes
```

```
- name: Создание и настройка Swarm кластера
```

```
hosts: managers
```

```
gather_facts: false
```

```
tasks:
```

```
- name: Инициализация Swarm кластера
```

```
command: docker swarm init
```

```
ignore_errors: true
```

```
register: swarm_init_output
```

```
changed_when: false
```

```
- debug:
```

```
var: swarm_init_output.stdout
```

```
- set_fact:
```

```
join_command: "{{ swarm_init_output.stdout_lines[-2] }}"
```

```
- name: Присоединение узлов к кластеру
```

```
hosts: nodes
```

```
gather_facts: false
```

```
tasks:
```

```
- name: Присоединение к кластеру
```

```
command: "{{ hostvars['manager'].join_command }}"
```

```
changed_when: false
```

```
ignore_errors: true
```

```
    register: swarm_join_output
- debug:
    var: swarm_join_output.stdout

# В зависимости от того добавляем или удаляем метки, нужно будет изменить `changed_when`
параметр заменить на true/false.
- name: Настройка меток для узлов кластера
  hosts: manager
  gather_facts: false

  tasks:
    - name: Добавление меток для узлов кластера
      command: docker node update --label-add {{ item.label }} {{ item.node }}
      with_items: "{{ node_labels }}"
      changed_when: false

    - name: Удаление меток для узлов кластера
      command: docker node update --label-rm {{ item.label }} {{ item.node }}
      with_items: "{{ node_labels }}"
      changed_when: false

  vars:
    node_labels:
      - { label: "env=prod", node: "node1" }
      - { label: "env=staging", node: "node2" }
      - { label: "env=dev", node: "node3" }
```

Команда для применения конфигурации с помощью ansible

```
ansible-playbook -i inventory.ini play-hostconf.yaml -kK
ansible-playbook -i inventory.ini play-glusterfs.yaml -kK
```

```
ansible-playbook -i inventory.ini play-docker.yaml -kK
```

Ключи -kK говорят о том, что я не использую id_rsa ключ. Вместо этого ввожу пароль для пользователя и для root доступа в ручную в момент запуска команды ansible-playbook. Если ваши хосты имеют установленные ключи, то можно опустить `-kK`.

Revision #3

Created 2023-10-11 11:48:16 UTC by gasick

Updated 2023-11-08 16:10:47 UTC by gasick