KUB 03: Развертывание Kubernetes-кластера с помощью kubespray

Описание:

Как мы уже говорили прошлом задании, существует довольно много, достаточно много вариантов для разворачивание self-hosted кластера kubernetes. И мы даже разобрали один из них - minikube. Сегодня же мы продолжим изучение установки kubernetes и остановимся на таком популярном инструменте как kubespray.
kubespray представляет из себя набор ansible ролей и плейбуков, которые позволят вам развернуть локальный kubernetes кластер. Данный способ можно считать одним из наиболее правильных для разворачивания кластера, поскольку все настройки описаны в системе управления конфигурациями, и вы можете вносить изменения в ваш кластер буквально одной командой. Помимо этого, kubespray содержит множество настроек, которые позволят вам привести ваш кластер в нужное состояние.
Давайте попробуем развернуть свой первый multi-node кластер с его помощью.
В первую очередь необходимо склонировать репозиторий и установить необходимые зависимости:

```
$ git clone https://github.com/kubernetes-sigs/kubespray
$ cd kubespray/
$ pip3 install -r requirements.txt
Defaulting to user installation because normal site-packages is
not writeable
... skipped ...
```

После этого необходимо составить свой inventory файл, который будет описывать наш кластер. Для примера мы можем скопировать шаблонный инвентори и изменить его:

```
$ cp -pr inventory/sample inventory/local
```

Внутри данного инвентори находится:

- group_vars - директория с переменными для разных типов хостов и установок
- inventory.ini - файл с описанием адресов и методов подключения по ssh

Давайте для начала изменим файл inventory.ini. Для этого в поставку kubespray входит питоновский скрипт, который сможет выполнить всю работу за нас:

```
# Объявляем IP адреса для нашего кластера
declare -a IPS=(134.122.85.85 134.122.69.63 161.35.28.90)
# запускаем скрипт генерации inventory:
```

```
CONFIG_FILE=inventory/local/hosts.yaml python3
contrib/inventory_builder/inventory.py ${IPS[@]}
```

В данном случае мы изменили файл с хостами на hosts.yaml, чтобы наш кластер был описан в yaml формате, а не в ini, как предлагается сделать по умолчанию. Давайте заглянем внутрь данного файла:

```
all:
  hosts:
    node1:
      ansible_host: 134.122.85.85
      ip: 134.122.85.85
      access_ip: 134.122.85.85
    node2:
      ansible_host: 134.122.69.63
      ip: 134.122.69.63
      access_ip: 134.122.69.63
    node3:
      ansible_host: 161.35.28.90
      ip: 161.35.28.90
      access_ip: 161.35.28.90
  children:
    kube_control_plane:
      hosts:
        node1:
        node2:
    kube-node:
      hosts:
        node1:
        node2:
        node3:
    etcd:
      hosts:
        node1:
        node2:
        node3:
    k8s-cluster:
      children:
        kube_control_plane:
        kube-node:
    calico-rr:
      hosts: {}
```

Как вы можете увидеть - по умолчанию скрипт описывает все ноды в all->hosts с их айпи адресами и другими параметрами. А ниже идет разделение нод по группам:

- kube_control_plane - группа мастер узлов - здесь будет запущен api-server, scheduler и controllers. Подробнее про kubernetes control plane поговорим в 7 задании.
- kube-node - группа узлов, на которых будет запускаться полезная нагрузка - то есть ваши приложения
- etcd - группа узлов, на которых будет запущен etcd кластер, который хранит в себе полное состояние кластера
- k8-cluster - объединение controle plane & kube-node хостов
- calico-rr - группа узлов route reflectors - относится к настройкам сети, о которых мы поговорим в 9 задании.

Итак, давайте немного исправим наш файл и сделаем первую ноду мастером (node1), а node2/3 - воркерами. А так же заставим etcd запускаться только на одной ноде. Конечно это не production ready инфраструктура, поскольку у нас отсутствует отказоустойчивость, но для тестов самое то. В идеале нужно иметь две или более мастер ноды и три etcd сервера для поддержания кворума. Итак, в итоге у нас получился вот такой файл:

```
all:
  hosts:
    node1:
      ansible_host: 134.122.85.85
      ip: 134.122.85.85
      access_ip: 134.122.85.85
    node2:
      ansible_host: 134.122.69.63
      ip: 134.122.69.63
      access_ip: 134.122.69.63
    node3:
      ansible_host: 161.35.28.90
      ip: 161.35.28.90
      access_ip: 161.35.28.90
  children:
    kube_control_plane:
      hosts:
        node1:
    kube-node:
      hosts:
        node2:
        node3:
    etcd:
      hosts:
```

```
    node1:
k8s-cluster:
  children:
    kube_control_plane:
    kube-node:
calico-rr:
  hosts: {}
```

Вторым шагом нам необходимо подправить переменные в group_vars и задать настройки нашего кластера. Заходим в group_vars/k8s-cluster и открываем файл k8s-cluster.yaml. Внутри вы найдете множество различных параметров. Давайте посмотрим на основные из них:

- kube_version: v1.20.5 - версия kubernetes, которую следует установить
- kube_network_plugin: calico - сетевой плагин, который будет использован для работы сети в кластере. Подробнее про сети мы поговорим в 9 задании. Для нас calico является хорошим стартовым выбором, так что не будем изменять.
- kube_service_addresses: 10.233.0.0/18 - сеть из которой будут выдаваться адреса для сервисов
- kube_pods_subnet: 10.233.64.0/18 - сеть из которой будут выдаваться адреса для подов
- kube_network_node_prefix: 24 - маска сети, которая будет выдаваться каждой ноде в кластере для выдачи адресов подам
- kube_apiserver_ip: "{{ kube_service_addresses|ipaddr('net')|ipaddr(1)|ipaddr('address') }}" - адрес на котором kubernetes api будет слушать входящие запросы
- kube_apiserver_port: 6443 - порт на котором будет работать kubernetes api
- cluster_name: cluster.local - имя зоны dns для нашего кластера
- dns_mode: coredns - режим работы днс. В данном случае мы деплоим в кластер coredns сервер, который будет отвечать за резолвинг имен внутри кластера
- enable_nodelocaldns: true - включаем кеширование dns запросов на каждой ноде в кластере - это означает, что контейнерам не придется каждый раз бегать к центральному coredns, расположенному на соседней машине
- container_manager: docker - container runtime - какой runtime для запуска контейнеров использовать - можно выбрать containerd, cri-o и docker.
- kubeconfig_localhost: true - генерировать или нет локальный kubeconfig для подключения к кластеру.

Ну собственно мы здесь ничего и не меняли, кроме kubeconfig_localhost, значение которого выставили в true. Так же советую обратить внимание на файл addons.yaml, в котором можно указать различные дополнения, которые вам может установить kubespray. Сейчас мы этого делать не будем, поскольку все дополнительные инструменты будем ставить руками.
После всех этих манипуляций мы можем попробовать запустить наш kube кластер:

```
$ ansible-playbook -i inventory/local/hosts.yaml -u root
cluster.yml
```

Я указал подключение под рутом к нашим серверам, поэтому нет смысла использовать sudo. После выполнения данной команды ваш кластер будет установлен примерно за 10 минут. Я советую сохранить вам данный плейбук, поскольку во всех заданиях перед его выполнением вам потребуется настраивать собственный кубер кластер (мы добавим дополнительные 30 минут к каждому заданию).
После установки давайте попробуем подключиться к нашему кластеру. Для этого можно использовать как локальный клиент kubectl, так и тот, который был установлен на мастер ноду. Давайте попробуем подключиться к мастер ноде и посмотреть на наш кластер:

```
root@node1:~# kubectl get nodes
NAME    STATUS   ROLES                  AGE     VERSION
node1   Ready    control-plane,master   5m25s   v1.20.5
node2   Ready    <none>                 3m50s   v1.20.5
node3   Ready    <none>                 4m3s    v1.20.5
```

Выглядит отлично! А подключиться локально можно, используя kubeconfig, который для нас сгенерировал kubespray:

```
$ KUBECONFIG=inventory/local/artifacts/admin.conf kubectl get
nodes
NAME    STATUS   ROLES                  AGE     VERSION
node1   Ready    control-plane,master   4h36m   v1.20.5
node2   Ready    <none>                 4h34m   v1.20.5
node3   Ready    <none>                 4h34m   v1.20.5
```

В данном случае, kubespray записал конфигурационный файл для подключения к кластеру в inventory/local/artifacts/admin.conf и мы указали его в качестве конфигурации для kubectl. Подробнее о том что находится внутри этого конфига и про утилиту kubectl мы поговорим в следующих заданиях.

Полезные ссылки:

- Kubespray (github)
- Getting started (official docs)
- Пример решения возможных проблем с python и required_pkgs

Задание:

1. Склонируйте репозиторий с kubespray локально на свою машину.
2. Выставите переменную для генерации локального kubeconfig.

3. Сгенерируйте inventory файл для вашего кластера:
- в control plane должны находиться две первые ноды (node1, node2)
- worker nodes и etcd должны входить все три ноды, которые были вам выданы (node1, node2, node3)
4. Разверните кластер Kubernetes с помощью kubespray
5. Отправьте задание на проверку.

P.S. Мы советуем сохранить директорию с kubespray и настроенным inventory, поскольку она еще пригодится вам для работы.