KUB 10: Kubernetes API

Описание:

Как мы уже упоминали в других заданиях, Kubernetes — это, по сути, API, который предоставляет доступ к разным ресурсам. И если вы создадите диск через API, то контроллер, который отвечает за создание дисков, увидит это, отправит запрос к провижионеру на создание диска и тот будет создан. Наше создание пода в прошлых заданиях — это было, по сути, создание ресурса в API. Давайте попробуем использовать API Kubernetes напрямую, без kubectl.
В нашем случае, kubespray генерирует нам клиентский ключ и сертификат, который можно использовать для доступа к кластеру. Давайте вытащим эти данные из kubeconfig и попробуем обратиться к API:

```
$ cat inventory/local/artifacts/admin.conf
apiVersion: v1
clusters:
- cluster:
    # Тут находится серверный CA сертификат
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJU...
    server: https://134.122.85.85:6443
  name: cluster.local
contexts:
- context:
    cluster: cluster.local
    user: kubernetes-admin-cluster.local
  name: kubernetes-admin-cluster.local@cluster.local
current-context: kubernetes-admin-cluster.local@cluster.local
kind: Config
preferences: {}
users:
- name: kubernetes-admin-cluster.local
  user:
    # Тут находится клиентский сертификат в base64
    client-certificate-data: LS0tLS1CRUdJTiBDRVJUSU
    # Тут находится клиентский ключ в base64
    client-key-data: LS0tLS1CRUdJTiBSU0EgUFJJVkFURS

# Создаем временную директорию
$ mkdir tmp
# Декодируем CA сертификат
```

```
$ echo 'LS0...' | base64 -d > tmp/ca.crt
# Декодируем клиентский сертификат
$ echo 'LS0...' | base64 -d > tmp/client.crt
# Декодируем клиентский ключ
$ echo 'LS0...' | base64 -d > tmp/client.key
```

После этого мы можем попробовать обратиться к API:

```
$ curl -D - -s --cacert tmp/ca.crt --cert tmp/client.crt --key
tmp/client.key https://134.122.85.85:6443/api
HTTP/2 200
cache-control: no-cache, private
content-type: application/json
x-kubernetes-pf-flowschema-uid:
0f28b092-9b8e-4749-8138-af2ebde847b6
x-kubernetes-pf-prioritylevel-uid:
69691e5a-7d4d-474c-8249-261d142b74b3
content-length: 185
date: Mon, 05 Apr 2021 22:48:19 GMT

{
  "kind": "APIVersions",
  "versions": [
    "v1"
  ],
  "serverAddressByClientCIDRs": [
    {
      "clientCIDR": "0.0.0.0/0",
      "serverAddress": "134.122.85.85:6443"
    }
  ]
}
```

Как видно из вывода - мы благополучно подключились к нашему Kubernetes API. Кстати, аутентифицироваться можно не только с помощью SSL сертификатов, но и с помощью token'ов, которые вам выдадут облачные утилиты (aws-cli, gcloud, etc...)

Kubernetes Standard API

Давайте теперь разберемся, где нам найти описание этого api. Во-первых, kubernetes описывает свое API, используя спецификацию openapi. Все API поделены на группы, у каждой группы есть версия и внутри этой версии есть ресурсы. Чтобы посмотреть все доступные группы, можно использовать kubectl:

```
$ kubectl api-resources
```

```
NAME                                SHORTNAMES   APIGROUP
NAMESPACED   KIND
bindings
true         Binding
componentstatuses                   cs
false        ComponentStatus
configmaps                          cm
true         ConfigMap
endpoints                           ep
true         Endpoints
events                              ev
true         Event
limitranges                         limits
true         LimitRange
namespaces                          ns
false        Namespace
nodes                               no
false        Node
persistentvolumeclaims              pvc
true         PersistentVolumeClaim
persistentvolumes                   pv
false        PersistentVolume
pods                                po
true         Pod
podtemplates
true         PodTemplate
replicationcontrollers              rc
true         ReplicationController
resourcequotas                      quota
true         ResourceQuota
secrets
true         Secret
serviceaccounts                     sa
true         ServiceAccount
services                            svc
true         Service
mutatingwebhookconfigurations
admissionregistration.k8s.io   false
MutatingWebhookConfiguration
```

```
validatingwebhookconfigurations
admissionregistration.k8s.io   false
ValidatingWebhookConfiguration
customresourcedefinitions         crd,crds
apiextensions.k8s.io           false
CustomResourceDefinition
apiservices
apiregistration.k8s.io         false        APIService
controllerrevisions                            apps
true         ControllerRevision
daemonsets                        ds           apps
true         DaemonSet
deployments                       deploy       apps
true         Deployment
replicasets                       rs           apps
true         ReplicaSet
statefulsets                      sts          apps
true         StatefulSet

... skipped ...
```

Также можно посмотреть доступные версии api для каждой группы, используя kubectl:

```
$ kubectl api-versions
admissionregistration.k8s.io/v1
admissionregistration.k8s.io/v1beta1
apiextensions.k8s.io/v1
apiextensions.k8s.io/v1beta1
apiregistration.k8s.io/v1
apiregistration.k8s.io/v1beta1
apps/v1
authentication.k8s.io/v1
authentication.k8s.io/v1beta1
authorization.k8s.io/v1
authorization.k8s.io/v1beta1
autoscaling/v1
autoscaling/v2beta1
autoscaling/v2beta2
batch/v1
batch/v1beta1
certificates.k8s.io/v1beta1
coordination.k8s.io/v1
```

```
coordination.k8s.io/v1beta1
discovery.k8s.io/v1beta1
events.k8s.io/v1beta1
extensions/v1beta1
metrics.k8s.io/v1beta1
networking.k8s.io/v1
networking.k8s.io/v1beta1
node.k8s.io/v1beta1
policy/v1beta1
rbac.authorization.k8s.io/v1
rbac.authorization.k8s.io/v1beta1
scheduling.k8s.io/v1
scheduling.k8s.io/v1alpha1
scheduling.k8s.io/v1beta1
snapshot.storage.k8s.io/v1beta1
storage.k8s.io/v1
storage.k8s.io/v1beta1
v1
```

Основные ресурсы сосредоточены в «пустой» группе — например, nodes, pods, events и так далее. Эта группа имеет только одну версию — v1, судя по выводу api-versions.
У группы apps также есть только одна версия — v1, и в ней сосредоточены ресурсы deployments, replicaset и другие.
Давайте попробуем посмотреть список нод в нашем кластере:

```
$ curl -D - -s --cacert tmp/ca.crt --cert tmp/client.crt --key
tmp/client.key https://134.122.85.85:6443/api/v1/nodes | head
-50
HTTP/2 200
cache-control: no-cache, private
content-type: application/json
x-kubernetes-pf-flowschema-uid:
0f28b092-9b8e-4749-8138-af2ebde847b6
x-kubernetes-pf-prioritylevel-uid:
69691e5a-7d4d-474c-8249-261d142b74b3
date: Mon, 05 Apr 2021 22:50:09 GMT

{
  "kind": "NodeList",
  "apiVersion": "v1",
  "metadata": {
    "resourceVersion": "945096"
```

```
  },
  "items": [
    {
      "metadata": {
        "name": "node1",
        "uid": "9958edd9-f571-4df0-b6ab-b851069a837f",
        "resourceVersion": "945090",
        "creationTimestamp": "2021-03-30T10:50:16Z",
        "labels": {
          "beta.kubernetes.io/arch": "amd64",
          "beta.kubernetes.io/os": "linux",
          "kubernetes.io/arch": "amd64",
          "kubernetes.io/hostname": "node1",
          "kubernetes.io/os": "linux",
          "node-role.kubernetes.io/control-plane": "",

          ... skipped ...
```

В таком случае мы использовали пустую группу (основную) с версией v1 и посмотрели ресурс nodes, так мы сконструировали url, к которому необходимо обратиться (/api//). Для остальных групп endpoint изменится на /apis. Давайте попробуем просмотреть список deployments, которые есть в данном кластере. Напомню, что группа для ресурса deployments — apps, а версия — v1:

```
$ curl -D - -s --cacert tmp/ca.crt --cert tmp/client.crt --key
tmp/client.key
https://134.122.85.85:6443/apis/apps/v1/deployments | head -50
HTTP/2 200
cache-control: no-cache, private
content-type: application/json
x-kubernetes-pf-flowschema-uid:
0f28b092-9b8e-4749-8138-af2ebde847b6
x-kubernetes-pf-prioritylevel-uid:
69691e5a-7d4d-474c-8249-261d142b74b3
date: Mon, 05 Apr 2021 22:50:43 GMT

{
  "kind": "DeploymentList",
  "apiVersion": "apps/v1",
  "metadata": {
    "resourceVersion": "945158"
  },
```

```
  "items": [
    {
      "metadata": {
        "name": "calico-kube-controllers",
        "namespace": "kube-system",
        "uid": "e5b12a43-d488-4c7b-92ba-e09d8ba0501e",
        "resourceVersion": "941926",
        "generation": 1,
        "creationTimestamp": "2021-03-30T10:53:00Z",
        "labels": {
          "k8s-app": "calico-kube-controllers"

          ... skipped ...
```

Давайте сейчас попробуем создать pod и посмотреть его через api:

```
$ kubectl run nginx --image nginx
pod/nginx created
$ curl -D - -s --cacert tmp/ca.crt --cert tmp/client.crt --key
tmp/client.key https://134.122.85.85:6443/api/v1/pods | head -50
HTTP/2 200
cache-control: no-cache, private
content-type: application/json
x-kubernetes-pf-flowschema-uid:
0f28b092-9b8e-4749-8138-af2ebde847b6
x-kubernetes-pf-prioritylevel-uid:
69691e5a-7d4d-474c-8249-261d142b74b3
date: Mon, 05 Apr 2021 22:51:20 GMT

{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "resourceVersion": "945236"
  },
  "items": [
    {
      "metadata": {
        "name": "nginx",
        "namespace": "default",
        "uid": "4b792b01-bf9a-4622-92f6-9652a1453767",
```

```
        "resourceVersion": "945225",
        "creationTimestamp": "2021-04-05T22:51:12Z",
        "labels": {
          "run": "nginx"
        },

        ... skipped ...
```

Собственно, создание ресурса происходит также через API — kubectl обращается к API Kubernetes с просьбой сделать новый ресурс, а после мы его просматриваем с использованием curl.
Понимание принципов построения API очень важно, поскольку мы будем создавать различные ресурсы в API.
Kubernetes Extended API
Если у вас есть желание — вы можете расширить API Kubernetes и добавить в него новые функции. Сделать это можно с помощью уже готового API apiregistration.k8s.io/v1, который дает возможность зарегистрировать ваш API в кластере Kubernetes. После этого вы сможете создавать собственные ресурсы, следить за ними и добавлять изменения в существующие объекты, чтобы реализовать собственную логику работы. Примерный yaml для регистрации выглядит следующим образом:

```
apiVersion: apiregistration.k8s.io/v1
kind: APIService
metadata:
  name: <name of the registration object>
spec:
  group: <API group name this extension apiserver hosts>
  version: <API version this extension apiserver hosts>
  groupPriorityMinimum: <priority this APIService for this
group, see API documentation>
  versionPriority: <prioritizes ordering of this version within
a group, see API documentation>
  service:
    namespace: <namespace of the extension apiserver service>
    name: <name of the extension apiserver service>
  caBundle: <pem encoded ca cert that signs the server cert used
by the webhook>
```

Предварительно вы должны запустить свой сервис внутри Kubernetes-кластера и уже после этого зарегистрировать его как расширение к API. Кстати, так регистрируется metrics-server, который отвечает за работу с метриками, использующимися в командах kubectl top. Можно посмотреть на него более подробно:

```
$ kubectl get apiservices
```

```
NAME                                     SERVICE
AVAILABLE   AGE

... skipped ..

v1beta1.authentication.k8s.io          Local
True        2d8h
v1beta1.authorization.k8s.io           Local
True        2d8h
v1beta1.batch                          Local
True        2d8h
v1beta1.certificates.k8s.io            Local
True        2d8h
v1beta1.coordination.k8s.io            Local
True        2d8h
v1beta1.discovery.k8s.io               Local
True        2d8h
v1beta1.events.k8s.io                  Local
True        2d8h
v1beta1.extensions                     Local
True        2d8h
v1beta1.metrics.k8s.io
kube-system/metrics-server   True        2d8h

... skipped ...

$ kubectl get apiservices v1beta1.metrics.k8s.io -o yaml
apiVersion: apiregistration.k8s.io/v1
kind: APIService

... skipped ...

spec:
  group: metrics.k8s.io
  groupPriorityMinimum: 100
  insecureSkipTLSVerify: true
  service:
    name: metrics-server
    namespace: kube-system
    port: 443
  version: v1beta1
```

```
  versionPriority: 100

... skipped ..
```

Полезные ссылки:

- Kubernetes API Deep Dive
- Kubernetes API overview

Задание:

1. Создайте pod с образом nginx и именем nginx-pod в namespace default.
2. Используя curl, добавьте поду label api=true.
3. Отправьте задание на проверку.