Kubernetes 配置管理中心 Secret

前言：

课程名称：Kubernetes 配置管理中心 Secret

实验环境：

本章节 Kubernetes 集群环境如下：

角色	IP	主机名	组件	硬件
控制节点	192.168.128.11	k8s-master01	apiserver controller-manager scheduler etcd containerd	CPU：4vCPU 硬盘：100G 内存：4GB 开启虚拟化
工作节点	192.168.128.21	k8s-node01	kubelet kube-proxy containerd calico coredns	CPU：6vCPU 硬盘：100G 内存：8GB 开启虚拟化
工作节点	192.168.128.22	k8s-node02	kubelet kube-proxy containerd calico coredns	CPU：6vCPU 硬盘：100G 内存：8GB 开启虚拟化

张岩峰老师微信，加我微信，邀请你加入 VIP 交流答疑群：

微信号：ZhangYanFeng0429

二维码：

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

1、初识 Secret

1.1、什么是 Secret？

Kubernetes（简称 K8s）Secret 是一种用于存储敏感数据的 Kubernetes API 对象。Secret 可以用于存储密码、密钥、API 令牌等敏感数据，并确保这些数据不被明文存储或传输。

Secret 可以以多种方式应用于 Kubernetes 部署中。例如，可以将 Secret 用于创建 Pod 和容器中的环境变量、命令行参数和卷。Secret 可以通过 Kubernetes API 创建，它可以存储加密的数据，并在需要使用敏感数据时将其解密。

Kubernetes Secret 通过使用加密和解密机制来确保敏感数据的安全。可以使用多种类型的 Secret，例如 Opaque Secret、TLS Secret、docker-registry Secret 等。通常，Secret 的使用是在 Kubernetes Pod 或 Deployment 中设置环境变量或挂载到容器中使用。

1.2、Secret 应用场景

Kubernetes Secret 的主要用途是保存和管理敏感信息，如密码、密钥、证书等。它可以应用于以下场景：

1、部署 Web 应用程序：你可以使用 Secret 来存储数据库密码、API 密钥、证书等信息，并将其作为环境变量注入到你的应用程序中。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

2、部署 SSL 证书：如果您需要使用 HTTPS 加密协议来保护 Web 应用程序的通信，您可以将 SSL 证书存储在一个 TLS 类型的 Secret 中，并将其挂载到 Pod 的卷中。

3、部署私有容器仓库：如果您需要访问您自己的私有 Docker 容器仓库，您可以使用 docker-registry 类型的 Secret 来存储 Docker 登录的用户名和密码，并将其作为环境变量注入到您的部署中。

4、部署第三方服务集成：如果你需要与第三方 API 集成，您可以使用 Secret 来存储 API 密钥和证书等敏感信息，并将其作为环境变量注入到你的应用程序中。这样可以保护你的 API 密钥和证书不被公开。

总之，Kubernetes Secret 提供一种方便的方法来管理和保护敏感信息，并允许您在部署中轻松地使用它们。

1.3、Secret 能够解决那些问题？

Kubernetes Secret 可以帮助解决以下问题：

1、安全性问题：使用 Secret 存储密码、证书、密钥等敏感信息可以避免这些信息被明文存储或传输，从而提高了部署的安全性。

2、环境管理问题：通过将敏感数据存储在 Secret 中，可以轻松地管理您的应用程序在不同环境中的配置，如开发、测试和生产环境。

3、与第三方服务集成问题：在访问和使用第三方 API 时，需要保护 API 密钥和证书等敏感信息，使用 Secret 可以方便地管理这些信息。

4、部署问题：在部署 Web 应用程序、私有容器仓库、SSL 证书等时，使用 Secret 可以简化配置，并保护敏感信息不暴露在配置文件中。

1.4、Secret 常用类型

在 Kubernetes 中，有以下五种主要类型的 Secret 对象：

1、Opaque：Opaque 类型是最常用的 Secret 类型，用于存储任意类型的数据。它可以存储任何格式的证书、密钥、密码等，但是不提供加密功能。

2、TLS：用于存储公钥、私钥和 CA 证书等用于 TLS 连接的数据。使用 TLS 类型的 Secret 可以解决部署 SSL 证书的问题。

3、Docker Registry：Docker Registry 类型用于在 Kubernetes 中设置私有 Docker Registry，存储用户名和密码等认证信息以便 Pod 中的容器使用。

4、Service Account：Service Account 类型 Secret 用于保存 Kubernetes 集群中的 Service Account Token 和 CA 证书信息，充当用于代表 Pod 的标识。

5、Generic：用于以 key-value 形式存储数据的 Secret，与 Opaque 类型类似。这种类型的 Secret 允许您使用不同的编码（base64、ascii）加密数据，但也不提供加密功能。

2、Secret 定义详解

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

Secret 资源可以通过如下命令查看相关语法：

```
[root@k8s-master01 ~]# kubectl explain Secret
```

- Secret 资源说明

属性名称	取值类型	取值说明
apiVersion	string	Api 版本
kind	string	资源类型
metadata	Object	元数据
metadata.name	String	控制器的名称
metadata.namespace	String	控制器所属的命名空间，默认值为 default
metadata.labels[]	List	自定义标签列表
data	map[string]string	用于存储 Secret 中的实际数据，这些数据通常是被 base64 编码的二进制数据。 在一个 Secret 对象中，可以定义多个键-值对，每个键值对都会被加密并保存在 data 字段中。
immutable	boolean	immutable 字段来指示该 Secret 在创建后是否可以被修改。如果将 Secret 的 immutable 字段设置为 true，则 Kubernetes 不允许修改或删除该 Secret。这可以确保在创建后没有人可以修改或删除这个 Secret，从而保证了数据安全性。
stringData	map[string]string	除了使用 data 字段存储 Secret 中的数据以外，还可以使用 stringData 字段。stringData 字段与 data 字段类似，都是用来存储 Secret 中的键值对。不同之处在于，stringData 字段中的数据不需要像 data 字段中的数据一样先进行 base64 编码。
type	string	type 字段指定了用于指定的 Secret 数据的编码和序列化格式。Kubernetes 中具有以下三种类型的 Secret 类型： ● Opaque：这是默认值，适用于任何类型的 Secret，包括二进制数据。 ● kubernetes.io/service-account-token：此类型使用 JWT 格式存储服务账户令牌和 CA 证书。 ● kubernetes.io/dockerconfigjson：此类型用于带有 Docker 认证令牌的 Secret。

3、Secret 常见用法

3.1、通过环境变量方式使用 Secret

在 Kubernetes 中，可以通过环境变量的方式使用 Secret。在 Pod 中，可以通过以下两种方式将 Secret 暴露给容器：

1、使用 envFrom 字段将 Secret 暴露给容器中的所有环境变量。

2、使用 valueFrom 字段将 Secret 暴露给容器中特定的环境变量。

下面分别介绍。

- 实战演练

  目标：将 mysql 数据库的登录名和密码做成 secret 挂载给 pod 容器使用

（1）创建 secret

```
1、创建资源清单文件
    [root@k8s-master01 ~]# vi mysql-user-pass.yaml
    apiVersion: v1
    kind: Secret
    metadata:
      name: mysql-user-pass
    type: Opaque
    data:
      username: root
      password: root

2、更新资源清单文件
    [root@k8s-master01 ~]# kubectl apply -f mysql-user-pass.yaml
    secret/mysql-user-pass created

3、查看 secret 资源
    [root@k8s-master01 ~]# kubectl get secret mysql-user-pass
    NAME              TYPE     DATA   AGE
    mysql-user-pass   Opaque   2      2m30s

4、查看 mysql-user-pass secret 资源的详细信息
    [root@k8s-master01 ~]# kubectl describe secret mysql-user-pass
    Name:         mysql-user-pass
    Namespace:    default
    Labels:       <none>
    Annotations:  <none>

    Type:  Opaque

    Data
    ====
    password:  3 bytes
    username:  3 bytes

5、查看 mysql-user-pass secret 资源的详细信息
    [root@k8s-master01 ~]# kubectl get secret mysql-user-pass -o yaml
```

```
[root@k8s-master01 ~]# kubectl get secret mysql-user-pass -o yaml
apiVersion: v1
data:
  password: root
  username: root
kind: Secret
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"password":"root","username":"root"},"kind":"Secret","metadata":{"annota
tions":{},"name":"mysql-user-pass","namespace":"default"},"type":"Opaque"}
  creationTimestamp: "2023-06-10T02:27:38Z"
  name: mysql-user-pass
  namespace: default
  resourceVersion: "16689"
  uid: f28d3d58-ecfb-49cf-977e-3fea326c7db7
type: Opaque
[root@k8s-master01 ~]#
```

通常情况下，data 字段不用于加密数据。Secret.data 在读取数据时，会将其解密。

我们的数据现在是没有加密的，当被挂载到 pod 容器中时，数据会进行解密，因为我们没有做数据加密，那么这时候数据会变成乱码。解决方法：1、使用 stringData 字段。2、对数据做加密。

这里我们先使用 stringData 字段。

6、更新资源清单文件

```
[root@k8s-master01 ~]# vi mysql-user-pass.yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysql-user-pass
type: Opaque
stringData:
  username: root
  password: root

[root@k8s-master01 ~]# kubectl apply -f mysql-user-pass.yaml
secret/mysql-user-pass configured
```

（2）以下是一个使用 envFrom 字段将 Secret 暴露给容器中的示例：

1、创建 secret 资源清单文件

```
[root@k8s-master01 ~]# vi pod-secret-1.yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-1
spec:
  containers:
    - name: myapp-1
      image: nginx:latest
      envFrom:
```

```
            - secretRef:
                name: mysql-user-pass
```

对上面的 yaml 文件说明：

envFrom 字段使用了 secretRef 字段来指定 Secret 的名称。这将会将 Secret 中的所有数据都暴露给容器的环境变量。

2、更新资源清单文件

```
[root@k8s-master01 ~]# kubectl apply -f pod-secret-1.yaml
pod/myapp-1 created
```

3、查看 pod

```
[root@k8s-master01 ~]# kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
myapp-1   1/1     Running   0          115s
```

4、进入容器，查看环境变量

```
[root@k8s-master01 ~]# kubectl exec -it myapp-1 -- bash
root@myapp-1:/# env | egrep "username|password"
username=root
password=root
```

从上面可以看出，我们在 secret 定义的 key 就是变量名，value 就是变量的值。

（3）如果只想将 Secret 中的某些值暴露给容器中的特定环境变量，可以使用 valueFrom 字段。以下是一个使用 valueFrom 字段将 Secret 暴露给容器中特定环境变量的示例：

1、创建 secret 资源清单文件

```
[root@k8s-master01 ~]# cat pod-secret-2.yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-2
spec:
  containers:
    - name: myapp-2
      image: nginx:latest
      env:
        - name: mysql_username
          valueFrom:
            secretKeyRef:
              name: mysql-user-pass
              key: username
```

```
            - name: mysql_password
              valueFrom:
                secretKeyRef:
                  name: mysql-user-pass
                  key: password
```

对上面的 yaml 文件说明：

valueFrom 字段使用 secretKeyRef 字段来指定 Secret 中的用户名和密码，然后将它们暴露给容器中的特定环境变量。

2、更新资源清单文件

```
[root@k8s-master01 ~]# kubectl apply -f pod-secret-2.yaml
pod/myapp-2 created
```

3、查看 pod

```
[root@k8s-master01 ~]# kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
myapp-2   1/1     Running   0          46s
```

4、进入容器，查看环境变量

```
[root@k8s-master01 ~]# kubectl exec -it myapp-2 -- bash
[root@k8s-master01 ~]# kubectl exec -it myapp-2 -- bash
root@myapp-2:/# env | egrep "username|password"
mysql_password=root
mysql_username=root
```

3.2、通过 volumeMount 使用 Secret

在 Kubernetes 中，可以使用 volumeMounts 字段指定将 Secret 对象挂载到 Pod 中的容器中。使用 volumeMounts 字段挂载 Secret 对象可以使 Pod 中的容器轻松地访问和使用 Secret 中的敏感数据。

实战：通过 volumeMount 使用 Secret

（1）创建 Secret

1、手动加密，基于 base64 加密

```
[root@k8s-master01 ~]# echo -n "root" | base64
cm9vdA==
[root@k8s-master01 ~]# echo -n "zz123.." | base64
enoxMjMuLg==
```

2、创建资源清单文件

```
[root@k8s-master01 ~]# vi secret-user-passwd.yaml
apiVersion: v1
```

```
    kind: Secret
    metadata:
      name: mysecret
    type: Opaque
    data:
      username: cm9vdA==
      password: enoxMjMuLg==
```

3、更新资源清单文件

```
    [root@k8s-master01 ~]# kubectl apply -f secret-user-passwd.yaml
    secret/mysecret created
```

4、查看 mysecret secret 资源详细信息

```
    [root@k8s-master01 ~]# kubectl get secret mysecret -o yaml
```

```
[root@k8s-master01 ~]# kubectl get secret mysecret -o yaml
apiVersion: v1
data:
  password: enoxMjMuLg==
  username: cm9vdA==
kind: Secret
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"password":"enoxMjMuLg==","username":"cm9vdA=="},"kind":"Secret","metada
ta":{"annotations":{},"name":"mysecret","namespace":"default"},"type":"Opaque"}
  creationTimestamp: "2023-06-10T04:21:35Z"
  name: mysecret
  namespace: default
  resourceVersion: "30461"
  uid: 1e7f7c29-87a0-489b-8e11-67c35746945b
type: Opaque
[root@k8s-master01 ~]#
```

5、解码

```
    [root@k8s-master01 ~]# echo "cm9vdA==" | base64 -d
    root
    [root@k8s-master01 ~]# echo "enoxMjMuLg==" | base64 -d
    zz123..
```

（2）创建 Pod 时使用 item 指定挂载 Secret 中的 key

当 Secret 资源中定义了多个 key 时，可以在创建 pod 时使用 item 指定 Secret 中的 key，挂载我们需要的加密文件。

1、创建资源清单文件

```
    [root@k8s-master01 ~]# vi secret-pod-1.yaml
    apiVersion: v1
    kind: Pod
    metadata:
      name: secret-pod-1
    spec:
      containers:
```

```
      - name: mycontainer
        image: nginx:latest
        volumeMounts:
          - name: mysecret-volume
            mountPath: /etc/mysecret
            readOnly: true
    volumes:
      - name: mysecret-volume
        secret:
          secretName: mysecret
          items:
          - key: username
            path: mysql-username.txt
          - key: password
            path: mysql-password.txt
```

对上面的 yaml 文件说明如下：

```
  apiVersion: v1
  kind: Pod
  metadata:
    name: secret-pod-1
  spec:
    containers:
      - name: mycontainer
        image: nginx:latest
        volumeMounts:
          - name: mysecret-volume    #指定 volume 的名称
            mountPath: /etc/mysecret  #将 volume 定义的内容,挂载到/etc/mysecret
目录下
            readOnly: true  #挂载到/etc/mysecret 目录下的文件只能读
    volumes:
      - name: mysecret-volume    #定义 volume 的名称为 mysecret-volume
        secret:
          secretName: mysecret  #使用 secret “mysecret”
          items:
          - key: username  #指定 secret “mysecret”的 key 为“username”，将此 key
的 value 挂载到 pod 内。
            path: mysql-username.txt  #当挂载到容器内时，叫什么文件名
          - key: password  #指定 secret “mysecret”的 key 为“password”，将此 key
的 value 挂载到 pod 内。
            path: mysql-password.txt  #当挂载到容器内时，叫什么文件名
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
2、更新资源清单文件
    [root@k8s-master01 ~]# kubectl apply -f secret-pod-1.yaml
    pod/secret-pod-1 created

3、查看 pod 资源
    [root@k8s-master01 ~]# kubectl get pods
    NAME           READY   STATUS    RESTARTS   AGE
    secret-pod-1   1/1     Running   0          3s

4、进入 pod，查看文件
    [root@k8s-master01 ~]# kubectl exec -it secret-pod-1 -- bash
    root@secret-pod-1:/# ls /etc/mysecret/
    mysql-password.txt  mysql-username.txt
    root@secret-pod-1:/# cat /etc/mysecret/mysql-username.txt
    root
    root@secret-pod-1:/# cat /etc/mysecret/mysql-password.txt
    zz123..
```

（3）创建 Pod 时不使用 item 指定挂载 secret 中的 key

在创建 pod 时如果不使用 item 指定 secret 中的 key，则当挂载到容器内时，会将 secret 中的 key 作为文件名、value 作为文件内容，全部挂载到容器内指的目录下。

```
1、创建资源清单文件
    [root@k8s-master01 ~]# vi secret-pod-2.yaml
    apiVersion: v1
    kind: Pod
    metadata:
      name: secret-pod-2
    spec:
      containers:
        - name: mycontainer
          image: nginx:latest
          volumeMounts:
            - name: mysecret-volume
              mountPath: /etc/mysecret
              readOnly: true
      volumes:
        - name: mysecret-volume
          secret:
            secretName: mysecret

2、更新资源清单文件
```

```
    [root@k8s-master01 ~]# kubectl apply -f secret-pod-2.yaml
    pod/secret-pod-2 created

3、查看 pod
    [root@k8s-master01 ~]# kubectl get pods secret-pod-2
    NAME           READY   STATUS    RESTARTS   AGE
    secret-pod-2   1/1     Running   0          34s

4、进入 pod，查看文件
    [root@k8s-master01 ~]# kubectl exec -it secret-pod-2 -- bash
    root@secret-pod-2:/# ls /etc/mysecret/
    password  username
    root@secret-pod-2:/# cat /etc/mysecret/username
    root
    root@secret-pod-2:/# cat /etc/mysecret/password
    zz123..
```

4、TLS Secret

下面是一个使用 Kubernetes TLS Secret 来保护应用程序通信的简单演练。

（1）创建证书和密钥

首先，在本地机器上创建证书和密钥。可以使用 openssl 命令或其他证书管理工具。在这里，我们使用以下命令来创建一对 RSA 密钥和自签名证书：

```
1、生成 CA 证书
    [root@harbor ~]# mkdir /data/ssl -p
    [root@harbor ~]# cd /data/ssl/

    # 生成一个 3072 位的 key，也就是私钥
    [root@harbor ssl]# openssl genrsa -out ca.key 3072

    # 生成一个数字证书 ca.pem，3650 表示证书的有效时间是 10 年
    [root@harbor ssl]# openssl req -new -x509 -days 3650 -key ca.key -out ca.pem
    You are about to be asked to enter information that will be incorporated
    into your certificate request.
    What you are about to enter is what is called a Distinguished Name or a DN.
    There are quite a few fields but you can leave some blank
    For some fields there will be a default value,
    If you enter '.', the field will be left blank.
    -----
    Country Name (2 letter code) [XX]:CN
    State or Province Name (full name) []:Beijing
```

```
    Locality Name (eg, city) [Default City]:Beijing
    Organization Name (eg, company) [Default Company Ltd]:system
    Organizational Unit Name (eg, section) []:CA
    Common Name (eg, your name or your server's hostname) []:harbor
    Email Address []:

2、查看证书文件
    [root@harbor ssl]# ll
    total 24
    -rw-r--r--. 1 root root 2455 May 22 23:16 ca.key
    -rw-r--r--. 1 root root 1647 May 22 23:17 ca.pem
```

（2）创建 TLS Secret

创建一个 Kubernetes 的 Secret 来存储 TLS 证书和密钥：

```
    [root@k8s-master01 ~]# kubectl create secret tls secret-tls
--cert=certs/ca.pem --key=certs/ca.key
    secret/secret-tls created
```

这里使用命令行进行创建 Secret，解释如下：

```
    这是一个用 kubectl 命令创建 TLS 证书密钥对的命令，将其保存在 Kubernetes 集群中的 secret 中。其中，secret-tls 是密钥对的名称，--cert 参数指定了证书文件的路径和名称，--key 参数指定了密钥文件的路径和名称。执行此命令需要保证证书和密钥文件路径正确，并且有足够的权限创建 secret 对象。
```

（3）创建 configmap 资源

这里创建一个 nginx-ssl.conf 配置文件，指定证书和私钥的位置。

```
1、创建资源清单文件
    [root@k8s-master01 ~]# vi myapp-cm.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: nginx-conf
    data:
      nginx-ssl.conf: |
        server {
            listen 443 ssl;
            server_name localhost;

            ssl_certificate /etc/tls/tls.crt;
            ssl_certificate_key /etc/tls/tls.key;
            ssl_session_timeout 5m;
            ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
            ssl_ciphers
```

```
ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;
            ssl_prefer_server_ciphers on;

            location / {
                root /usr/share/nginx/html;
                index index.html index.htm;
            }
        }

2、更新资源清单文件
    [root@k8s-master01 ~]# kubectl apply -f myapp-cm.yaml
    configmap/nginx-conf created
```

（4）创建 deployment

```
1、创建 deployment 资源清单文件
    [root@k8s-master01 ~]# vi myapp-tls.yaml
    apiVersion: apps/v1
    kind: Deployment
    metadata:
      name: myapp-tls
      labels:
        app: myapp-tls
    spec:
      replicas: 1
      selector:
        matchLabels:
          app: myapp-tls
      template:
        metadata:
          labels:
            app: myapp-tls
        spec:
          containers:
          - name: nginx
            image: nginx
            ports:
            - containerPort: 443
            volumeMounts:
            - name: tls-certs
              mountPath: /etc/tls
            - name: nginxconf
              mountPath: /etc/nginx/conf.d/nginx-ssl.conf
```

```
            subPath: nginx-ssl.conf
        volumes:
        - name: tls-certs
          secret:
            secretName: secret-tls
        - name: nginxconf
          configMap:
            name: nginx-conf
```

2、更新资源清单文件

```
[root@k8s-master01 ~]# kubectl apply -f myapp-tls.yaml
deployment.apps/myapp-tls created
```

3、查看 pod

```
[root@k8s-master01 ~]# kubectl get pods -o wide
```

```
[root@k8s-master01 ~]# kubectl get pods -o wide
NAME                         READY   STATUS    RESTARTS   AGE     IP              NODE         NOMINATED NODE   READINESS GATES
myapp-tls-7ddfd75c56-7mln4   1/1     Running   0          4m48s   10.244.58.204   k8s-node02   <none>           <none>
[root@k8s-master01 ~]#
```

3、访问 pod，测试

```
[root@k8s-master01 ~]# curl https://10.244.58.204:443 -k
```

```
[root@k8s-master01 ~]# curl https://10.244.58.204:443 -k
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[root@k8s-master01 ~]#
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。