Day - 10

1) Configmap

2) Secret

1) Configmap :-

- It is an API object that used to store non-Sensetive data which is of key:value pair.

→ pods can consume configmap as

1) env variables.

2) CLI Arguements

3) Configuration files in the volume.

- Most of kubernetes objects have "Specs" but configmap has "data" and "binarydata".
- The name of configmap should be valid DNS Sub domain.

There are four different way that you can use configmap to configure a container inside a pod.

i) Inside a container commands and args.

ii) Environment variables.

iii) Add a file in read-only mode for application to read.

iv) Write a code to run inside the pod that uses kubernetes to read the configmap.

→ Checkout the below example for Configmap as Kubernetes objects.

```
apiVersion : v1
kind : Configmap
metadata :
    name : game-demo
data :
    player_initial_lives : "3"
    ui-properties-file-name : "user_interface.properties"
game.properties : |
    enemy.type = aliens, mastes
    player.maximum-lives = 5
user-interface.properties : |
    Color.good = purple
    color.bad = yellow
    allow.textmode = true.
```

For more details on configmap please visit the official Kubernetes documentation, and Search Config - Explore more details there.

2) Secrets :-

- It is an Kubernetes object which contains small amount of sensitive data such as password access key and token.
- We can use these info in pods specs but the usage of Secrets will enable more Secured way of using of password, keys--- and also we don't need to use of confidential info in application code.

→ Both Configmap and Secrets are of same type except the usage varies here.

→ Secrets will be stored in etcd by default by kubernetes API, so those who has access to APIs will have access to Secrets too.

→ To avoid such usage of unathorized access.

1) We should enable encryption to Secrets at rest.

2) Enable RBAC access to Secrets.

3) Consider usage of external providers to Store Secrets.

4) Restrict the access to specific Containers (Secrets)

→ Uses of Secrets:-

1) Used to Store Environment variables.

2) To Store SSH keys or passwords to pods.

3) Allow Kubelet to pull images from private registries.

* There are various Alternatives for Secrets that you can checkout @ kubernetes documentation.

* There are various types of Secrets that you can Specify in the Secret. As per the official documentation you can checkout below.

Built-in type	Usage
→ opaque	→ arbitrary user-defined data
→ kubernetes.io/service-account-token	→ ServiceAccount token
→ kubernetes.io/dockercfg	→ Serialized ~/.dockercfg file
→ kubernetes.io/dockerconfigjson	→ Serialized ~/.docker/config.json file
→ kubernetes.io/basic-auth	→ Credentials for basic authentication
→ kubernetes.io/ssh-auth.	→ credentials for SSH authentication.
→ kubernetes.io/tls	→ data for a TLS client or server.
→ bootstrap.kubernetes.io/token	→ bootstap token data

All these are built-in secret types you can use.

Here, I have illustrated a basic Secret of type ssh key:

```
apiVersion: v1
kind: Secret
metadata:
    name: secret-ssh-auth
type: kubernetes.io/ssh-auth
data:
  ssh-privatekey: |
    UG91cmluZzRWIVdGIjb24lU2NIYME=
```

The fields in the secrets file will be varied according to the use-case that you are using.

- There are various type to create secrets such as.
  1) Kubectl command.
  2) Configuration / definition file
  3) using kustomize tool

you can edit the secrets will all three methods but using kustomize tool it will generate new secrets with updated data.

"Access to secrets will be according to the least privillage manner" (most important in realtime)

Every use case secrets and detailed into about each case are fabricated at kubernetes / secrets page. Just have look around to know more about them.