#KubernetesIn30Days challenge :-

#Day10 :-

① Config Map

② Secret

① Config Map :-

→ It is an API object that used to store Non-sensitive 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 K8 objects have "specs" but configmap has "data" and "binaryData"

→ The name of Config Map should be valid DNS sub domain.

There are four different ways that you can use ConfigMap to configure a container inside a Pod

① Inside a container commands and args

② Environment variables

③ Add a file in read-only mode for application to read.

④ Write a code to run inside the Pod that uses K8 API to read the configMap.

checkout the below example for ConfigMap as Kubernetes object.

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: game-demo
data:
  # property-like keys; each key maps to a simple value
  player_initial_lives: "3"
  ui_properties_file_name: "user-interface.properties"

  # file-like keys
  game.properties: |
    enemy.types=aliens,monsters
    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 ConfigMap. - Explore more details there.

② Secrets :-

→ It is an Kubernetes object which contains small amount of sensitive data such as password, access keys and token.

→ We can use these info in Pod's specs but the usage of secrets will enable more secured way of using passwords, keys ... and also we don't need to use of confidential info in application code.

→ Both configMap & 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 unauthorized access

① We should enable Encryption to secrets at rest
② Enable RBAC access to secrets.
③ consider usage of external provider 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 @ K8 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`	bootstrap token data

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

Here, I have illustrated a basic secret of type SSH key:

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: secret-ssh-auth
type: kubernetes.io/ssh-auth
data:
  # the data is abbreviated in this example
  ssh-privatekey: |
    UG91cmluZzYlRW1vdGljb24lU2N1YmE=
```

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

There are various ways to create secrets. such as

1. Kubectl command
2. configuration/definition file
3. Using Kustomize Tool

You can Edit the secrets with all three methods but using Kustomize Tool it'll generate new secret with updated data

"Access to secrets will be according to the least privillage manner"
(Most Important in realtime)

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