

The Ondia logo is centered on a white background with purple corner accents. It features the word "ondia" in a lowercase, rounded sans-serif font. The letter "o" is purple, while "ndia" is a darker blue. A small teal and blue geometric shape is positioned above the "i".

ondia

The slide features a large illustration on the right side depicting a team of people interacting with a large laptop displaying charts and graphs. Several small figures are shown around the laptop, some sitting on the screen, others standing and pointing at the display. A magnifying glass is also present. The entire scene is rendered in a clean, isometric style with a color palette of purples, blues, and oranges. In the top right corner, there is a small circular logo with a blue and teal design. The title "Kubernetes Secrets and ConfigMaps" is written in a blue sans-serif font, preceded by a purple triangle. The Ondia logo is in the bottom left corner, followed by a horizontal line and a small blue circle with a white dot inside.

Kubernetes Secrets and ConfigMaps

ondia

AGENDA

- ▶ **Secrets**
- ▶ **ConfigMaps**

Secrets

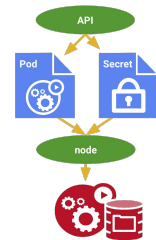
1

Why env exist?

```
1 API_KEY="XXX_YYY"
2 PASSWORD="mypassword"
3 HOST="111.121.130.17"
4 PORT="3360"
5 DATABASE="mydb"
6 USER="Mickey"
```

Secrets

A **Secret** is an object that contains a small amount of **sensitive data such as a password, a token, or a key**. Such information might otherwise be put in a Pod specification or in an image.



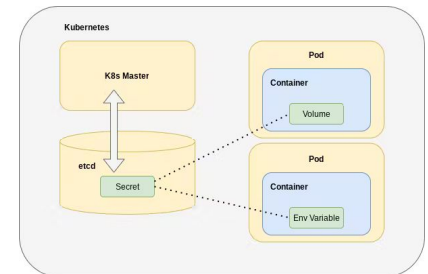
Secrets

Opaque is the default Secret type if omitted from a Secret configuration file. When you create a Secret using kubectl, you will use the **generic** subcommand to indicate an Opaque Secret type.

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

Secrets

Secrets can be mounted as data **volumes** or exposed as **environment variables** to be used by a container in a Pod.

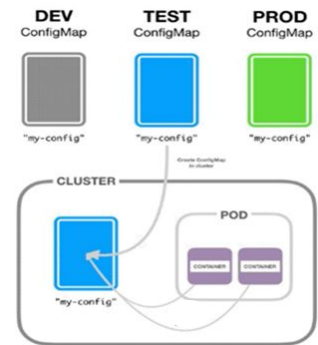


ConfigMaps



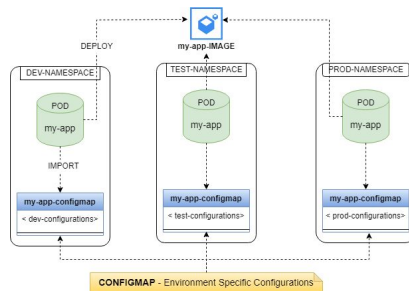
ConfigMaps

- A **ConfigMap** is an API object used to store **non-confidential data in key-value pairs**. Pods can consume ConfigMaps as **environment variables**, **command-line arguments**, or as **configuration files in a volume**.



ConfigMaps

- A ConfigMap allows you to **decouple environment-specific configuration** from your container images, so that your applications are easily portable.



```
env:  
  - name: APP_COLOR  
    value: pink
```

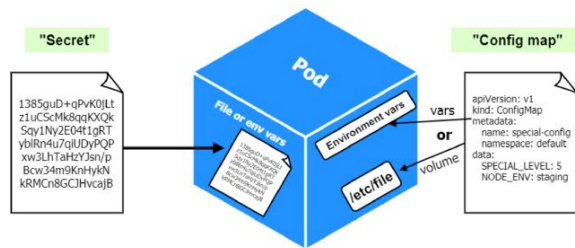
1 Plain Key Value

```
env:  
  - name: APP_COLOR  
    valueFrom:  
      configMapKeyRef:
```

2 ConfigMap

```
env:  
  - name: APP_COLOR  
    valueFrom:  
      secretKeyRef:
```

3 Secrets



ConfigMaps Samples

> User can create configMap via **Literal** or **from Files**.

> **Via File:** A **path to a directory** containing one or more configuration files, indicated using the **—from-file** flag.

*kubectl create configmap [NAME] --from-file [/PATH/TO/FILE.PROPERTIES]
--from-file [/PATH/TO/FILE2.PROPERTIES]*

> User can also put complete directory, containing multiple files.

kubectl create configmap [NAME] --from-file [/PATH/TO/DIRECTORY]

ConfigMaps Samples

- > **Via Literal Values:** To create a ConfigMap from literal values
—from-literal.

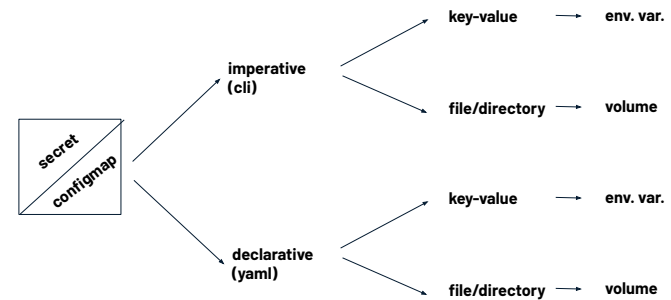
```
kubectl create configmap literal-data --from-literal key1=value1 --  
from-literal key2=value2
```

```
kubectl create configmap special-config --from-literal=special.how=very  
--from-literal=special.type=charm
```

- > Get ConfigMap via CLI.

```
kubectl get configmaps <config-map_Name> -o yaml/json
```

Handson Workflow





THANKS!
Any questions?



Control Plane Components

