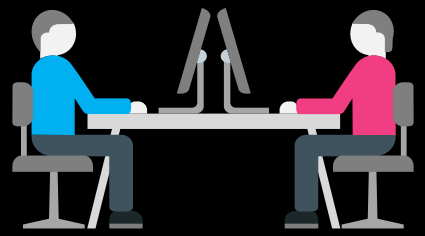
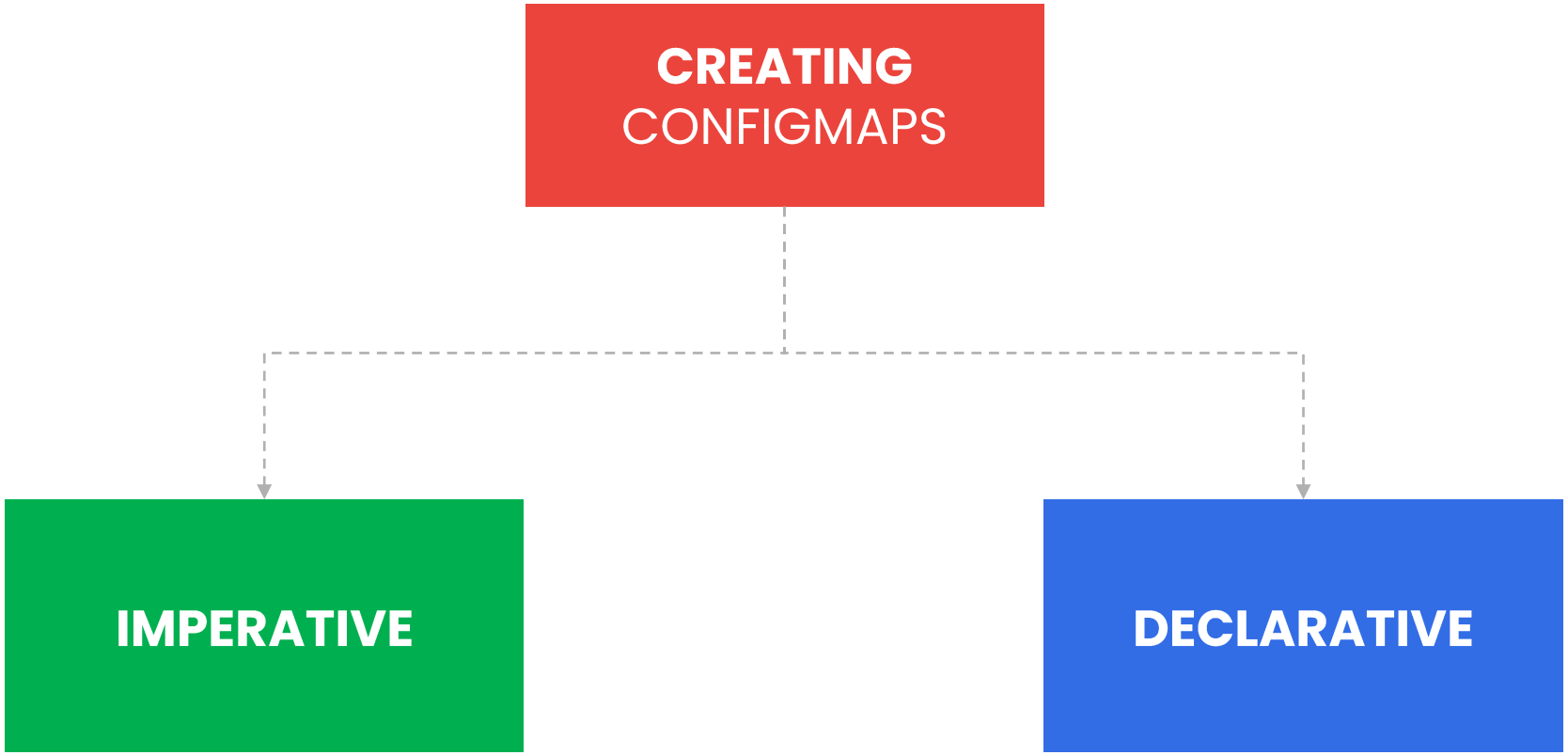
Creating Configmap



KUBERNETES: **API-SERVER** OVERVIEW



```
ENV_ONE: value1
ENV_TWO: value2
```

CREATING CONFIGMAPS USING IMPERATIVE METHOD

SYNTAX:

```
kubectl create configmap [MAP-NAME] [DATA SOURCE]
```

FROM LITERAL VALUES

```
kubectl create configmap special-config --from-literal=ENV_ONE=value1 --from-literal=ENV_TWO=value2
configmap/special-config created
```

CREATING CONFIGMAPS USING IMPERATIVE METHOD

configmaps-file.txt

```
This is configmap example
```

SYNTAX:

```
kubectl create configmap [MAP-NAME] [DATA SOURCE]
```

FROM LITERAL VALUES

```
kubectl create configmap special-config --from-literal=ENV_ONE=value1 --from-literal=ENV_TWO=value2
configmap/special-config created
```

FROM FILE

```
kubectl create configmap special-config --from-file=/path/to/configmap-file.txt
configmap/special-config created
```

```
kubectl create configmap my-config --from-file=path/to/bar
configmap/special-config created
```

CREATING CONFIGMAPS USING DECLARATIVE METHOD

cat special-config.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: special-config
data:
  ENV_ONE: “va1ue1”
  ENV_TWO: “va1ue2”
```

cat nginx-config.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-nginx-config
data:
  my-nginx-config.conf: |-
    server {
      listen 80;
      server_name www.kubia-example.com;
      gzip on;
      gzip_types text/plain application/xml;
      location / {
        root /usr/share/nginx/html;
        index index.html index.htm;
      }
    }
    sleep-interval: 25
```

File Name - KEY

File Content - VALUE