Creating a configmap

A ConfigMap in Kubernetes is an API object used to store non-confidential configuration data in key-value pairs. It allows you to decouple configuration artifacts from container images, making your applications more portable and easier to configure.

What Can You Store in a ConfigMap?

ConfigMaps can store:

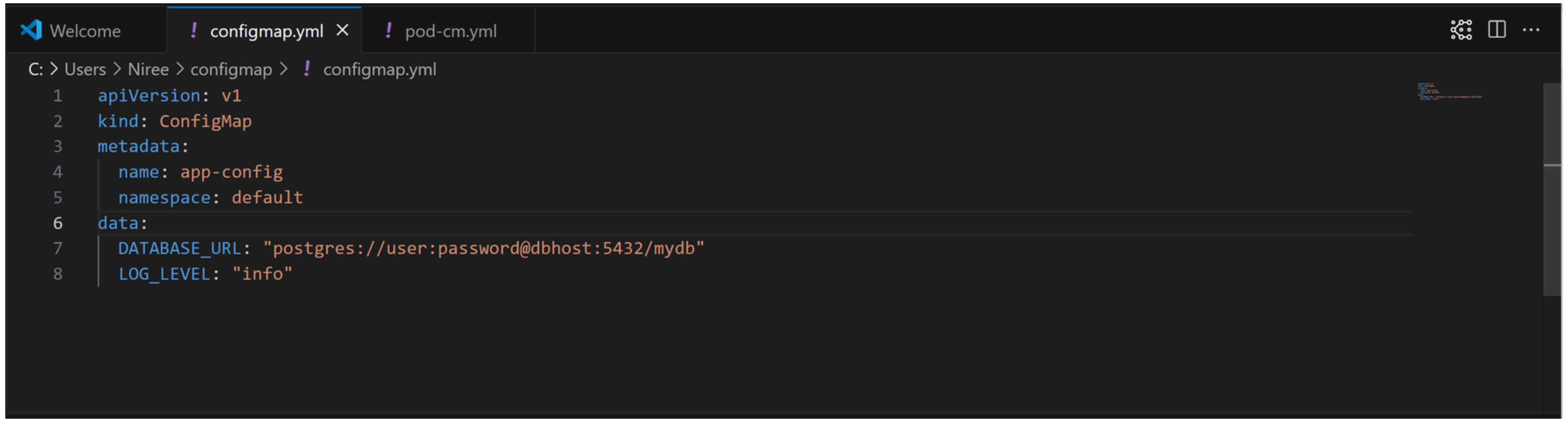
Key-value pairs (e.g., environment variables).

Configuration files (e.g., .properties, .json, .xml).

Command-line arguments.

Any plain text data.

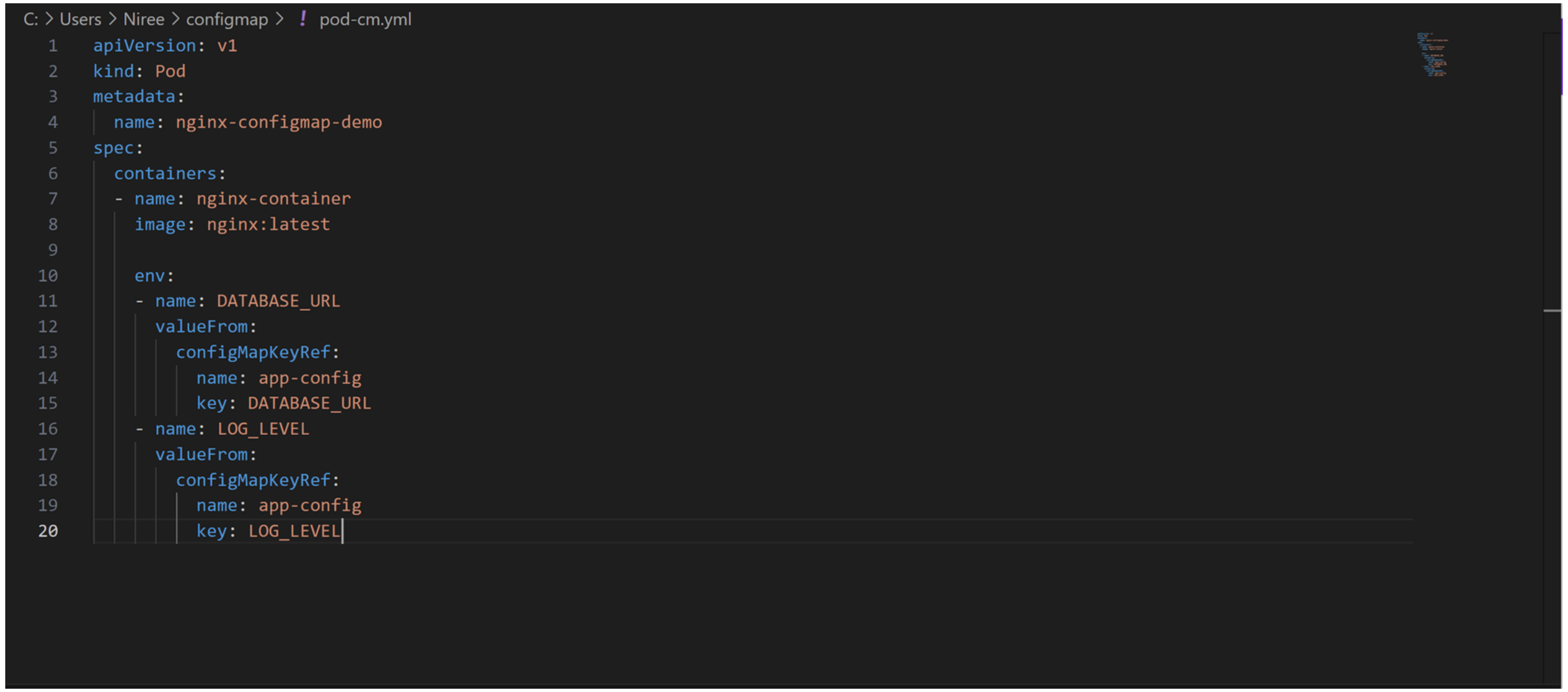
Created Config .yml file



```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
  namespace: default
data:
  DATABASE_URL: "postgres://user:password@dbhost:5432/mydb"
  LOG_LEVEL: "info"
```

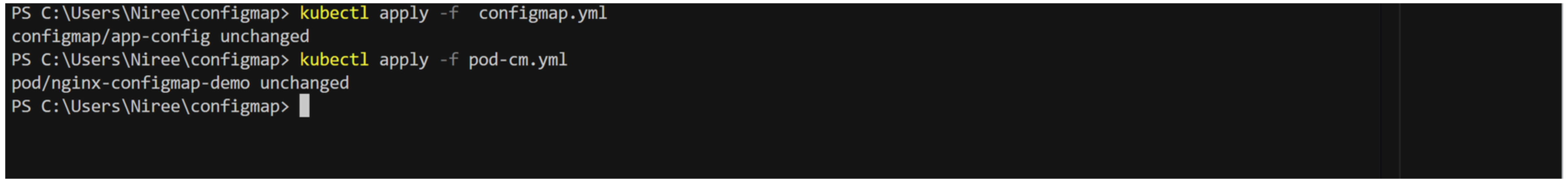
First Create Configmap.yml file
then specify the name and give data

Created pod.yml file


C: > Users > Niree > configmap > ! pod-cm.yml

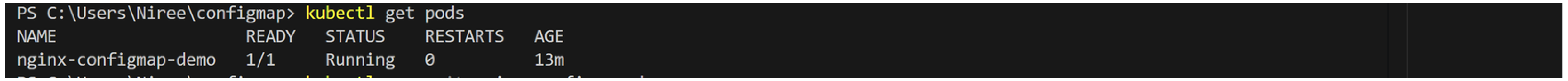
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-configmap-demo
spec:
  containers:
  - name: nginx-container
    image: nginx:latest

    env:
    - name: DATABASE_URL
      valueFrom:
        configMapKeyRef:
          name: app-config
          key: DATABASE_URL
    - name: LOG_LEVEL
      valueFrom:
        configMapKeyRef:
          name: app-config
          key: LOG_LEVEL
```

Create Pod.yml file with nginx container and add envi details like database url and log-level



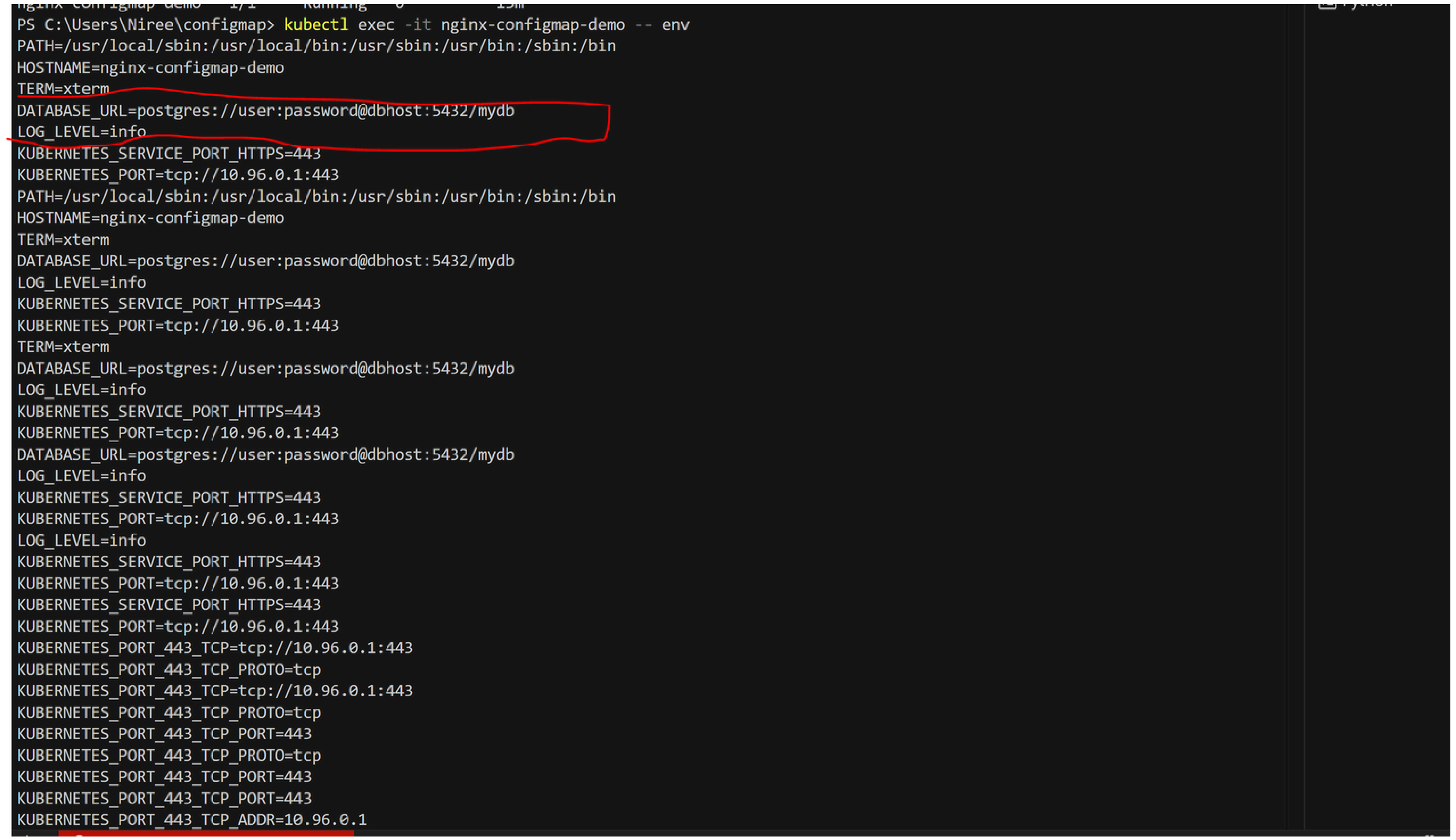
```
PS C:\Users\Niree\configmap> kubectl apply -f  configmap.yml
configmap/app-config unchanged
PS C:\Users\Niree\configmap> kubectl apply -f pod-cm.yml
pod/nginx-configmap-demo unchanged
PS C:\Users\Niree\configmap>
```

Apply configmap.yml file and pod_cm.yml file



```
PS C:\Users\Niree\configmap> kubectl get pods
NAME                   READY   STATUS    RESTARTS   AGE
nginx-configmap-demo   1/1     Running   0          13m
```

Then Check Pod is running are not



```
PS C:\Users\Niree\configmap> kubectl exec -it nginx-configmap-demo -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=nginx-configmap-demo
TERM=xterm
DATABASE_URL=postgres://user:password@dbhost:5432/mydb
LOG_LEVEL=info
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=nginx-configmap-demo
TERM=xterm
DATABASE_URL=postgres://user:password@dbhost:5432/mydb
LOG_LEVEL=info
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
TERM=xterm
DATABASE_URL=postgres://user:password@dbhost:5432/mydb
LOG_LEVEL=info
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
DATABASE_URL=postgres://user:password@dbhost:5432/mydb
LOG_LEVEL=info
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
LOG_LEVEL=info
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
```

With help of **kubectl exec -it nginx-configmap-demo -- env** command check envi are added in pod

configmap

- First Create configmap.yml and pod.yml
- Then on configmap.yml file write a data
- ex (data-base-url & log file)
- Then pod.yml file write a envidata
- After that apply both file
- Then check pod and configmap
- Finally check where all envi are added in pod with help of this command **kubectl exec -it nginx-configmap-demo -- env**