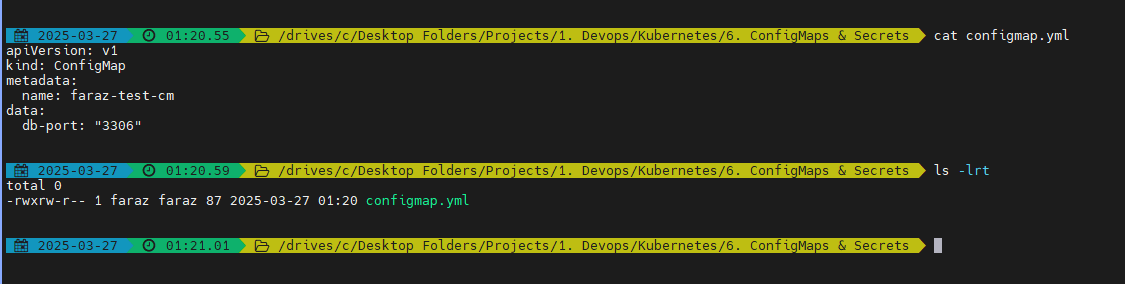
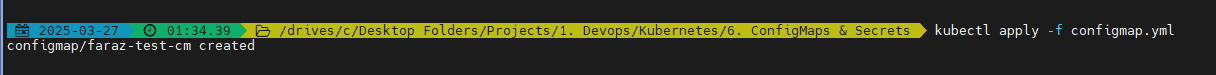
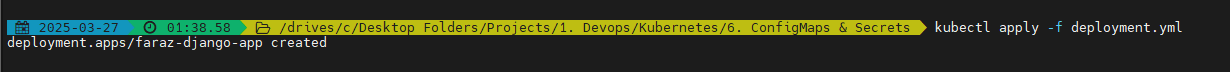
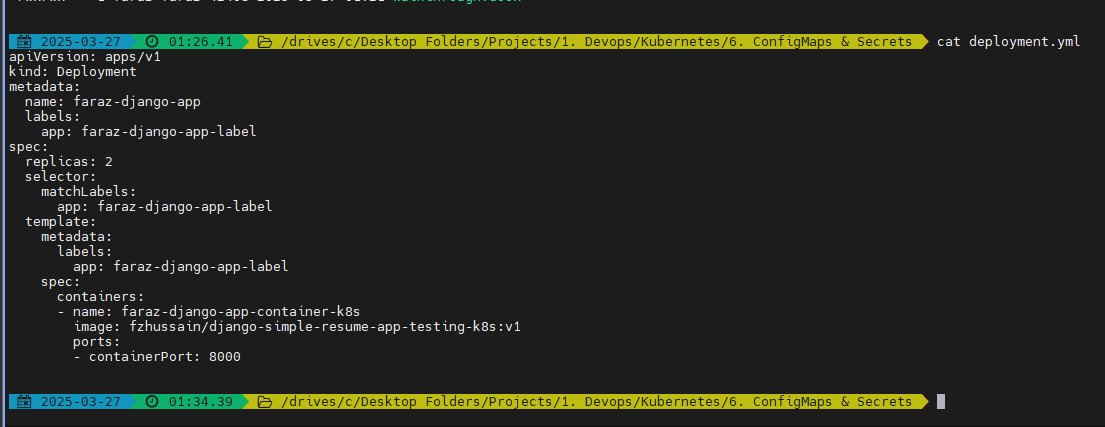
Create your config map:

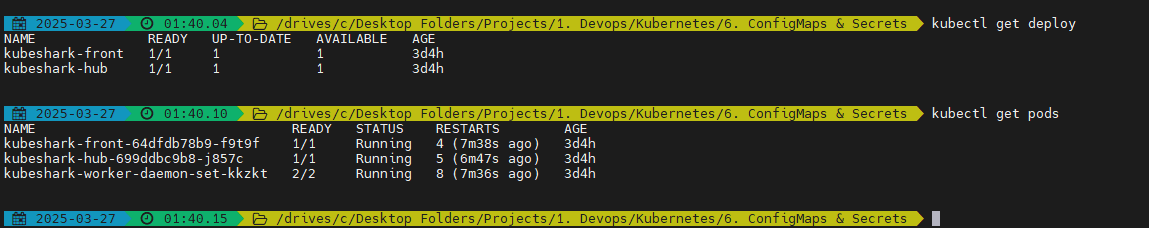


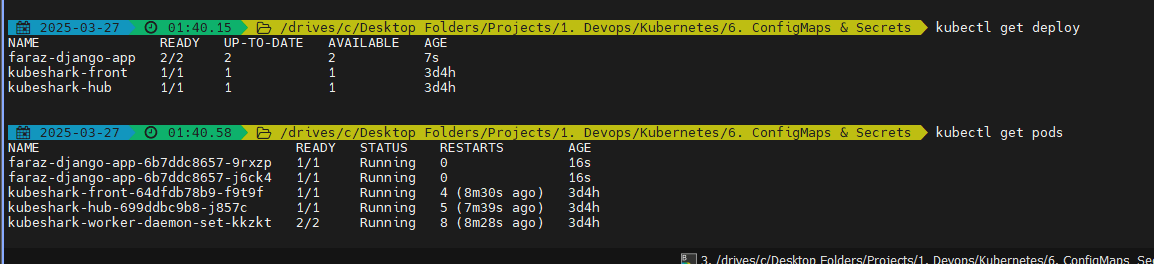




We may edit the same deployment that we used earlier:



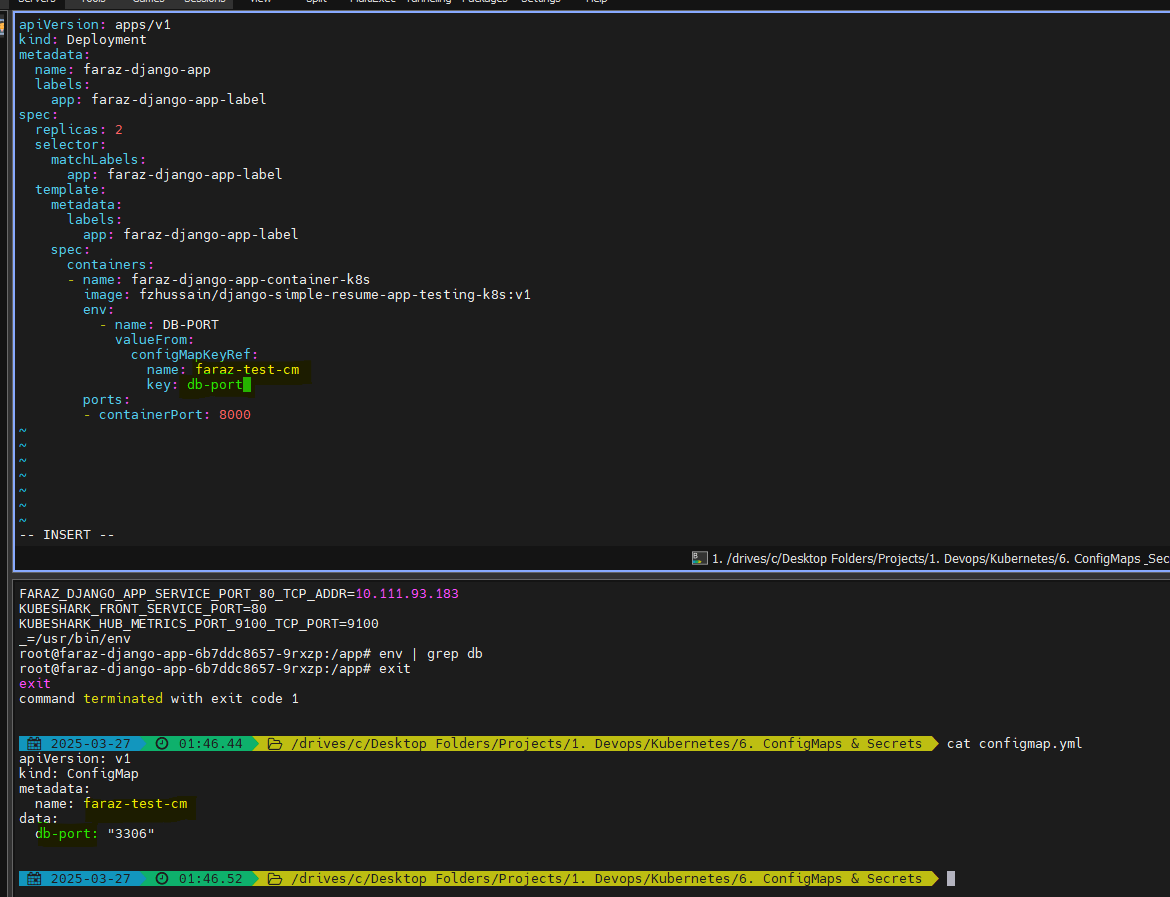




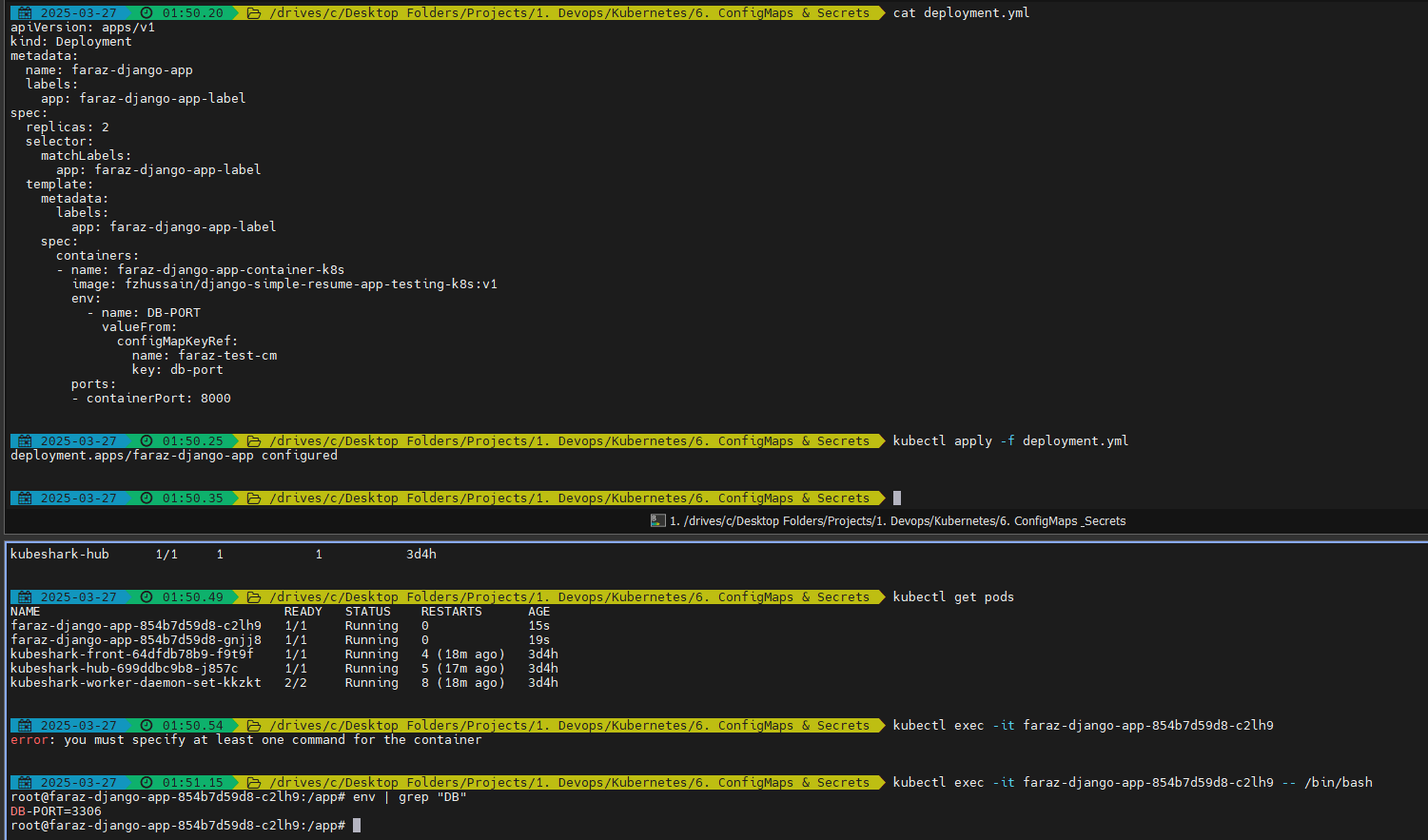
We will be able to see no environment variable is seen:



Create your deployment:

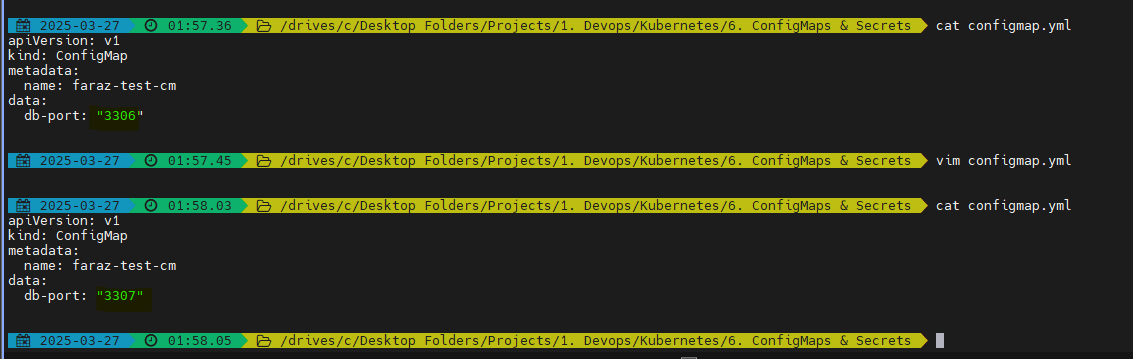


Apply the deployment:

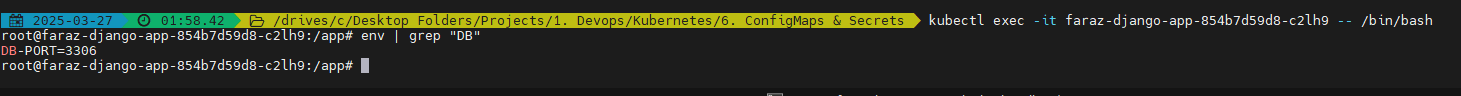


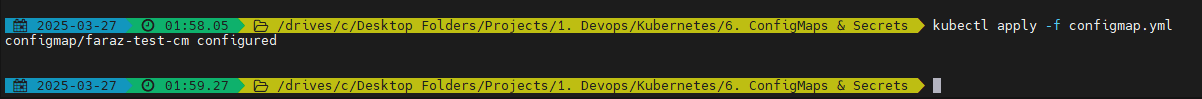
Now from here developer can use this env variable

Now for some reason if I want to change the value of the environment variable:

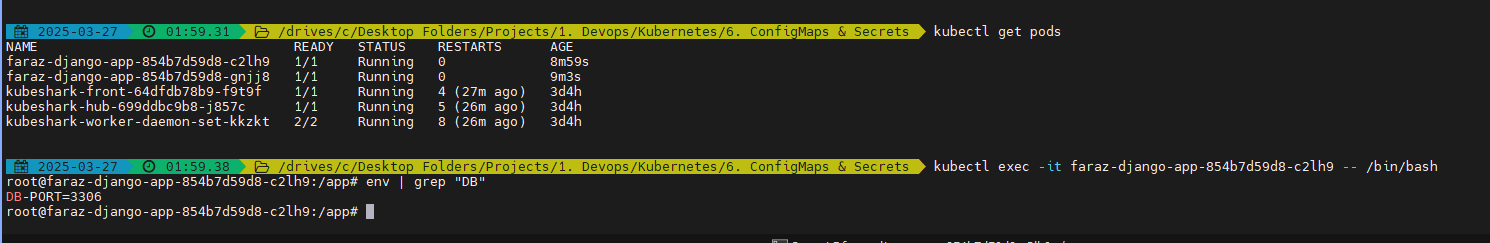


The env variable is not changed:





As you can see even after applying the changes the pods were not updated:

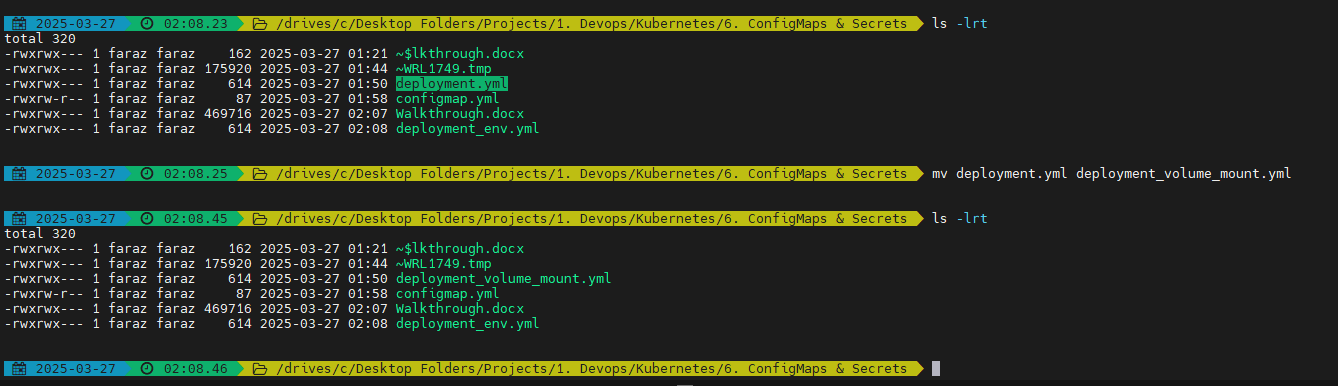


The updations of values inside container/pod is not possible in K8s as restarting the deployment can lead to traffic loss

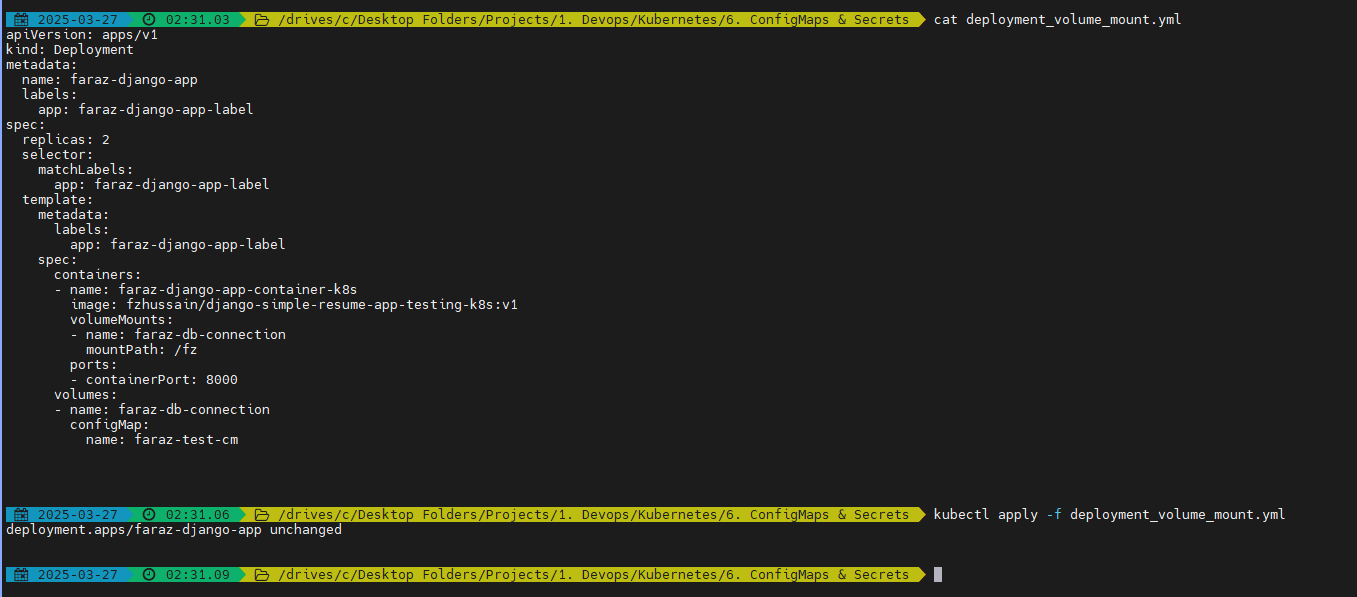
To solve this problem we use Volume mounts

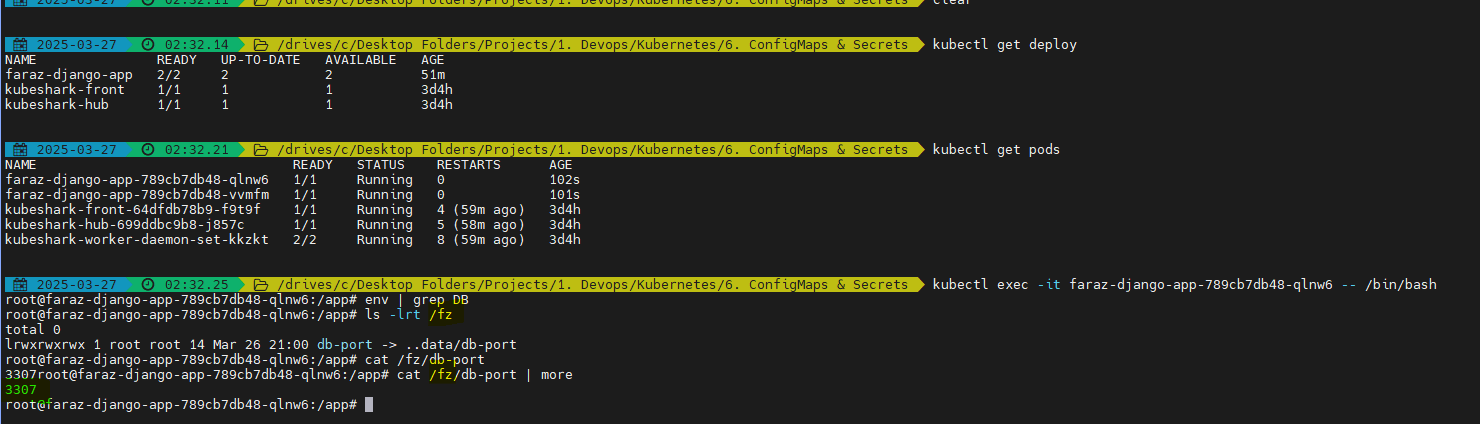
* So instead of env variables we use files

Create a copy for future reference:

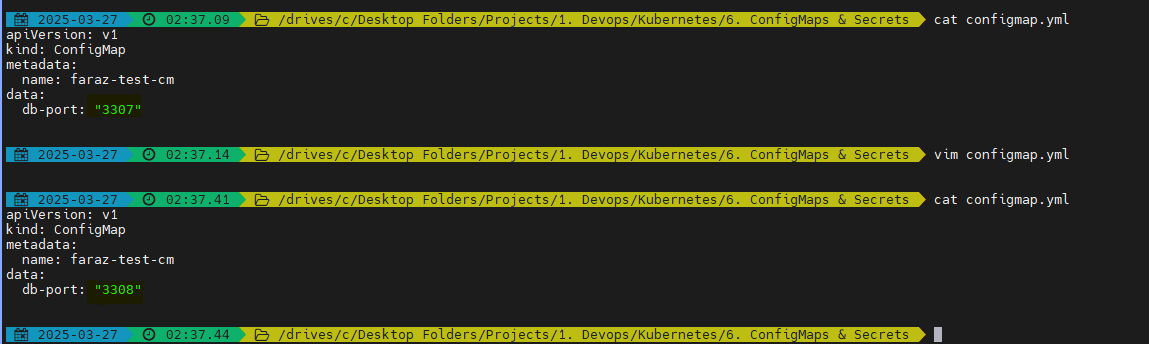


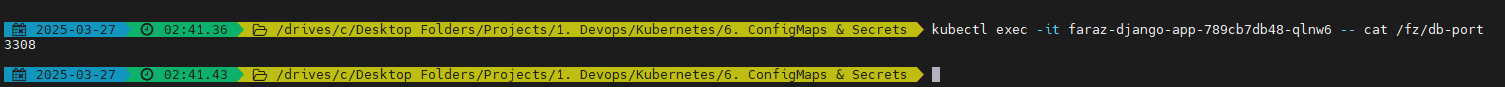
Create your deployment:



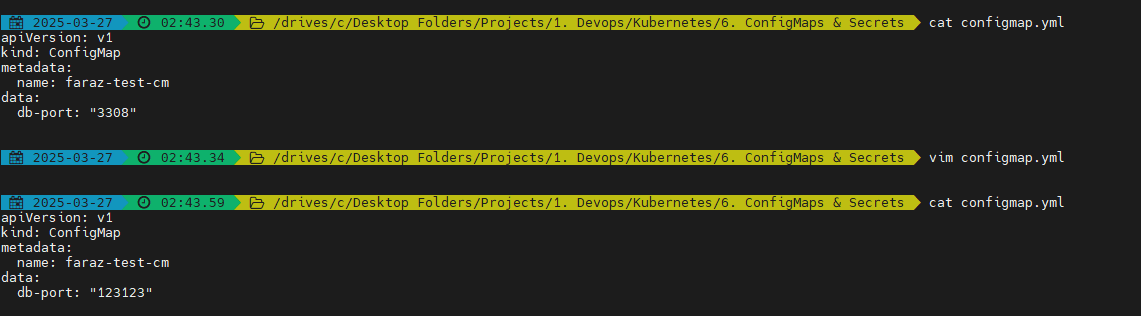


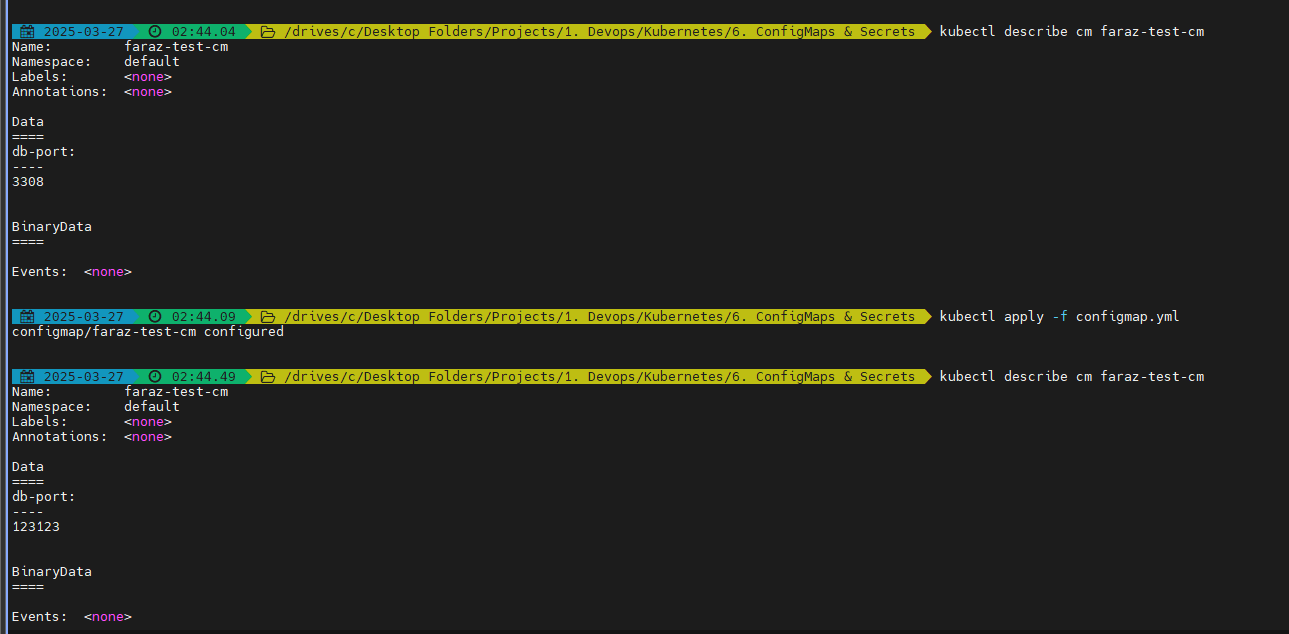
Now if I just edit my config map the details inside fz/db-port must be changed



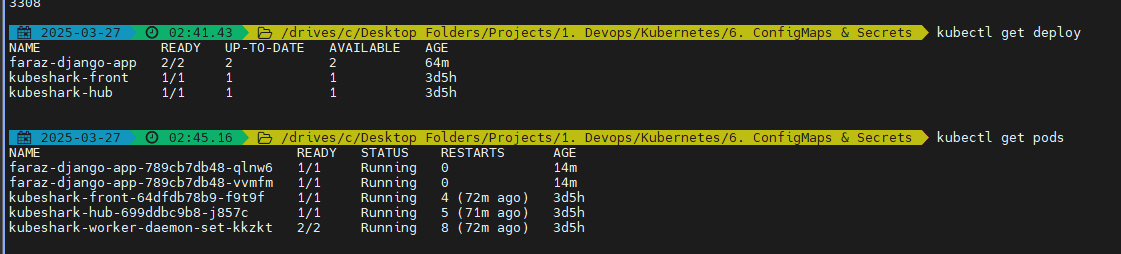


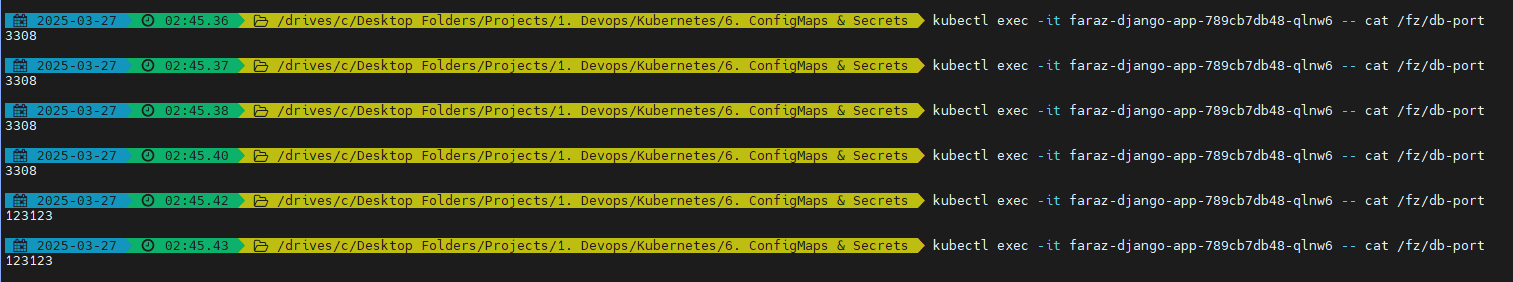
Let’s demonstrate one more time:





As you can see the pods are not refreshed:

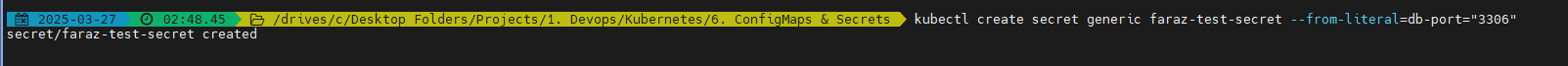


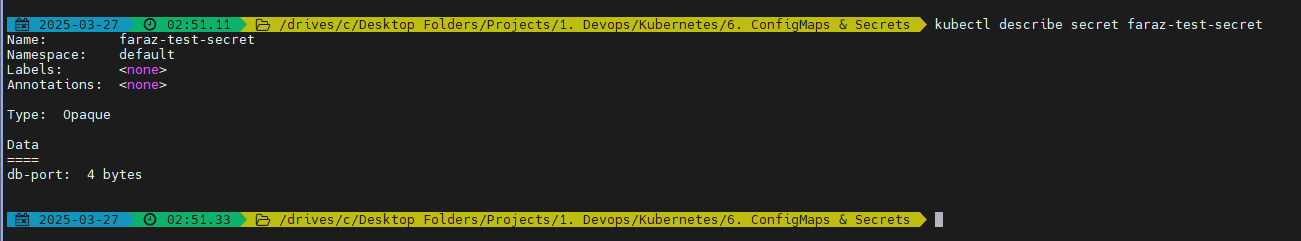
After sometime the details get updated:

Now let’s create secrets:

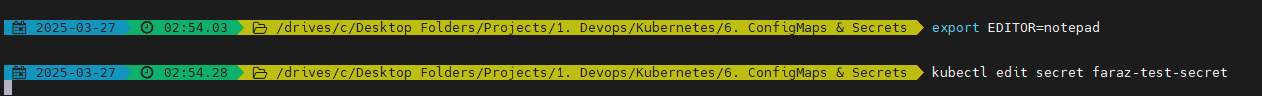
We can either use the YAML approach or make secrets/config maps/ deployments using:

* kubectl create secret generic faraz-test-secret --from-literal=db-port="3306"

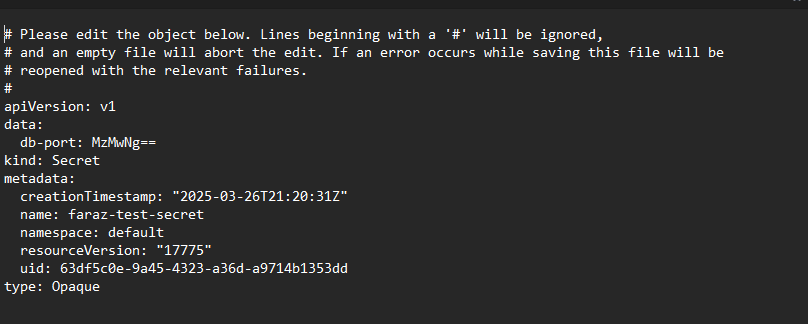




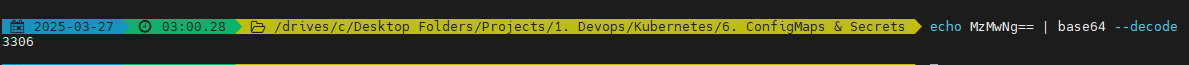
* kubectl edit secret faraz-test-secret



* Secrets encrypts the data



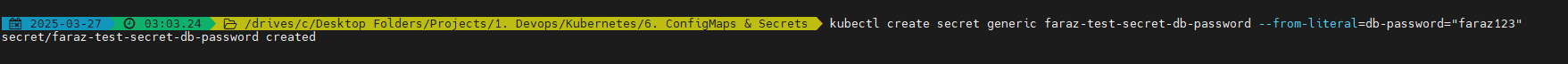
* Decode the data using base64:
  + echo MzMwNg== | base64 --decode



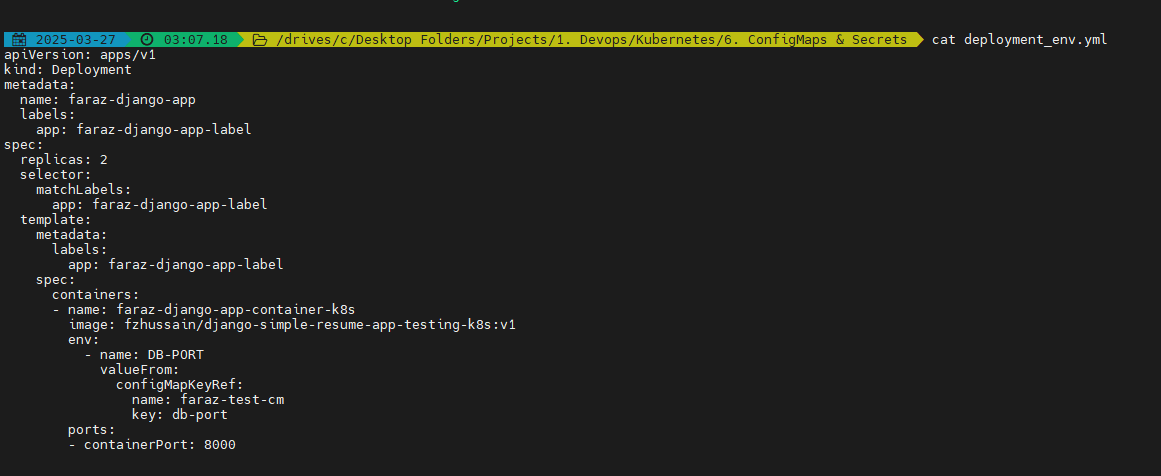
Always use encryption key at ETCD for secrets

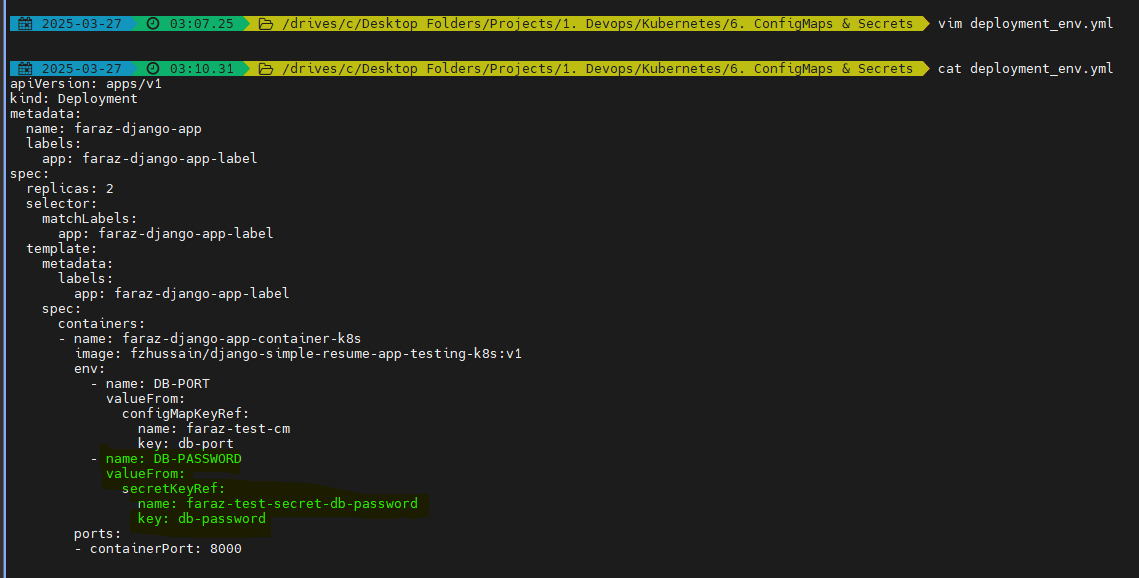
Creating a new secret:

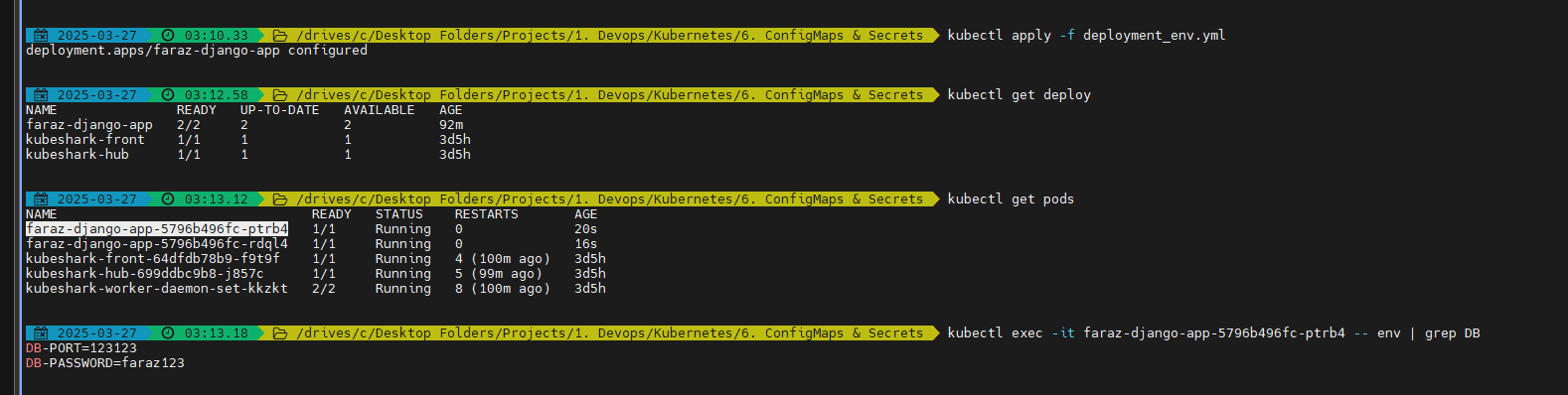
* kubectl create secret generic faraz-test-secret-db-password --from-literal=db-password="faraz123"



Earlier:







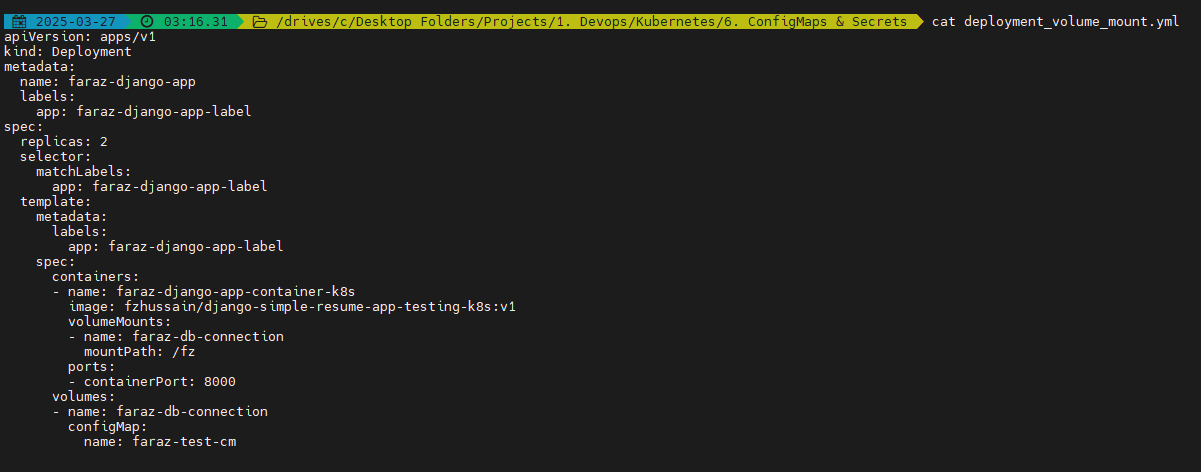
When creating a secret like faraz-test-secret-db-password, Kubernetes stores the db-password value as a Base64-encoded string.

When injected into the container (via environment variables or files), Kubernetes decodes the Base64 automatically and makes the raw secret value available in the container.

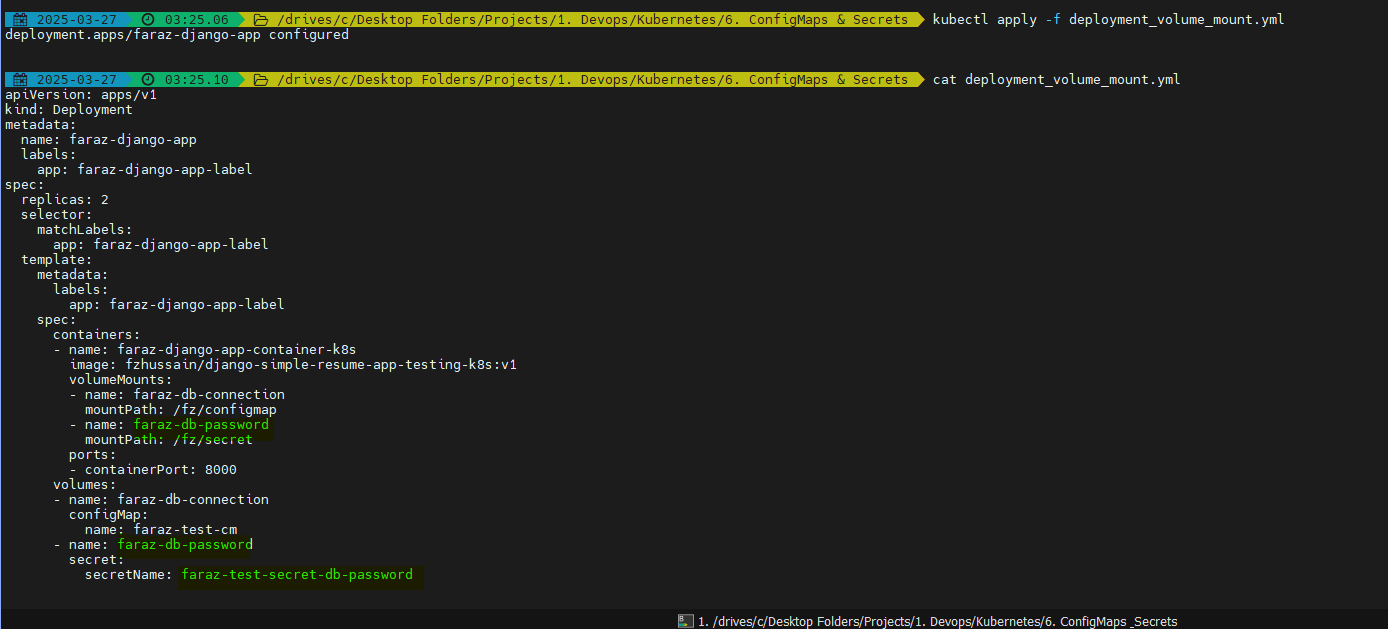
As a result, when you access DB-PASSWORD via kubectl exec, you see the decoded raw value (faraz123 in your case).

Now for the case of volume mounts:

Earlier:

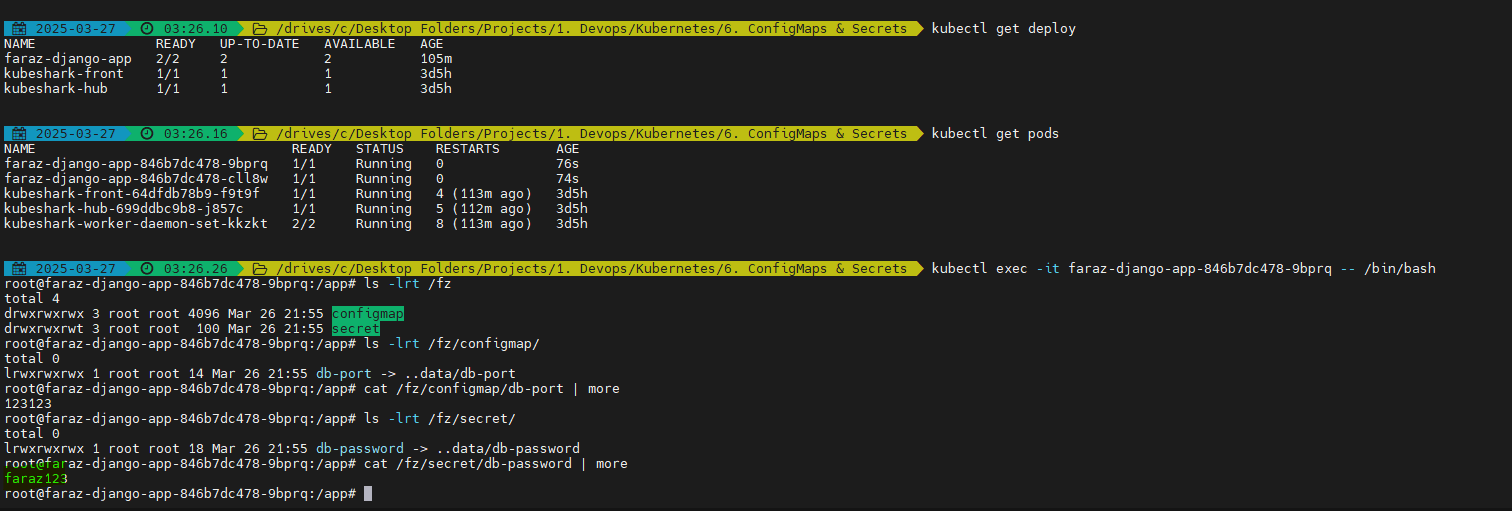


Current:

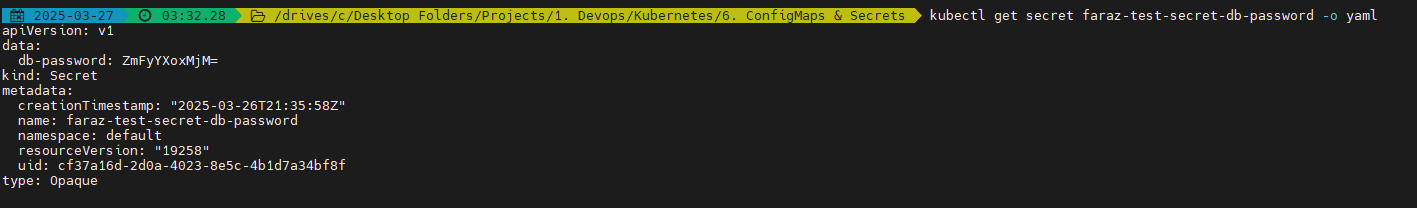


You will be able to view the secret on as shown below:

* kubectl exec -it faraz-django-app-846b7dc478-9bprq -- /bin/bash



* kubectl get secrets
* kubectl get secret faraz-test-secret-db-password -o yaml



* echo ZmFyYXoxMjM= | base64 --decode

