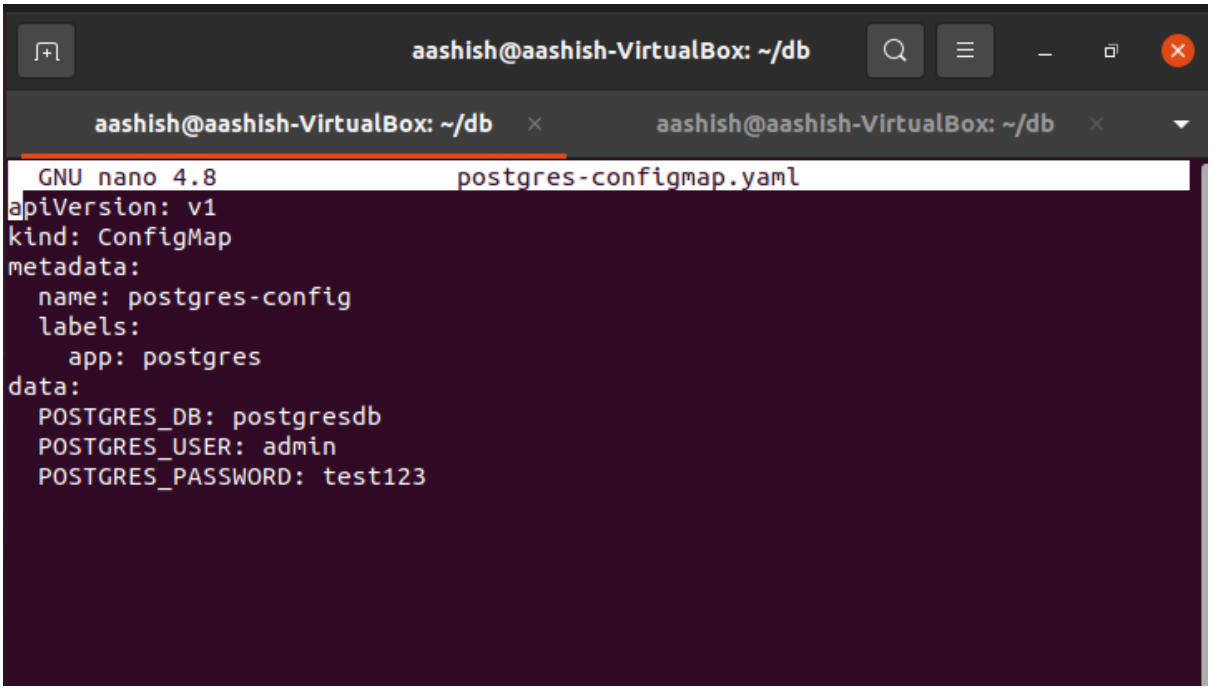
**1.**
**Deploy Postgres database using PVC & PV cluster**

**Answer:**

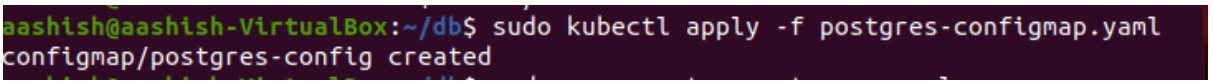
I have created a directory named db to store all the files for Postgres database.
The ConfigMap resource contains the data that is used during the deployment process.
So, first of all, we create a ConfigMap YAML file named **postgres-configmap.yaml** in a text editor as follows;

- **sudo nano postgres-configmap.yaml**

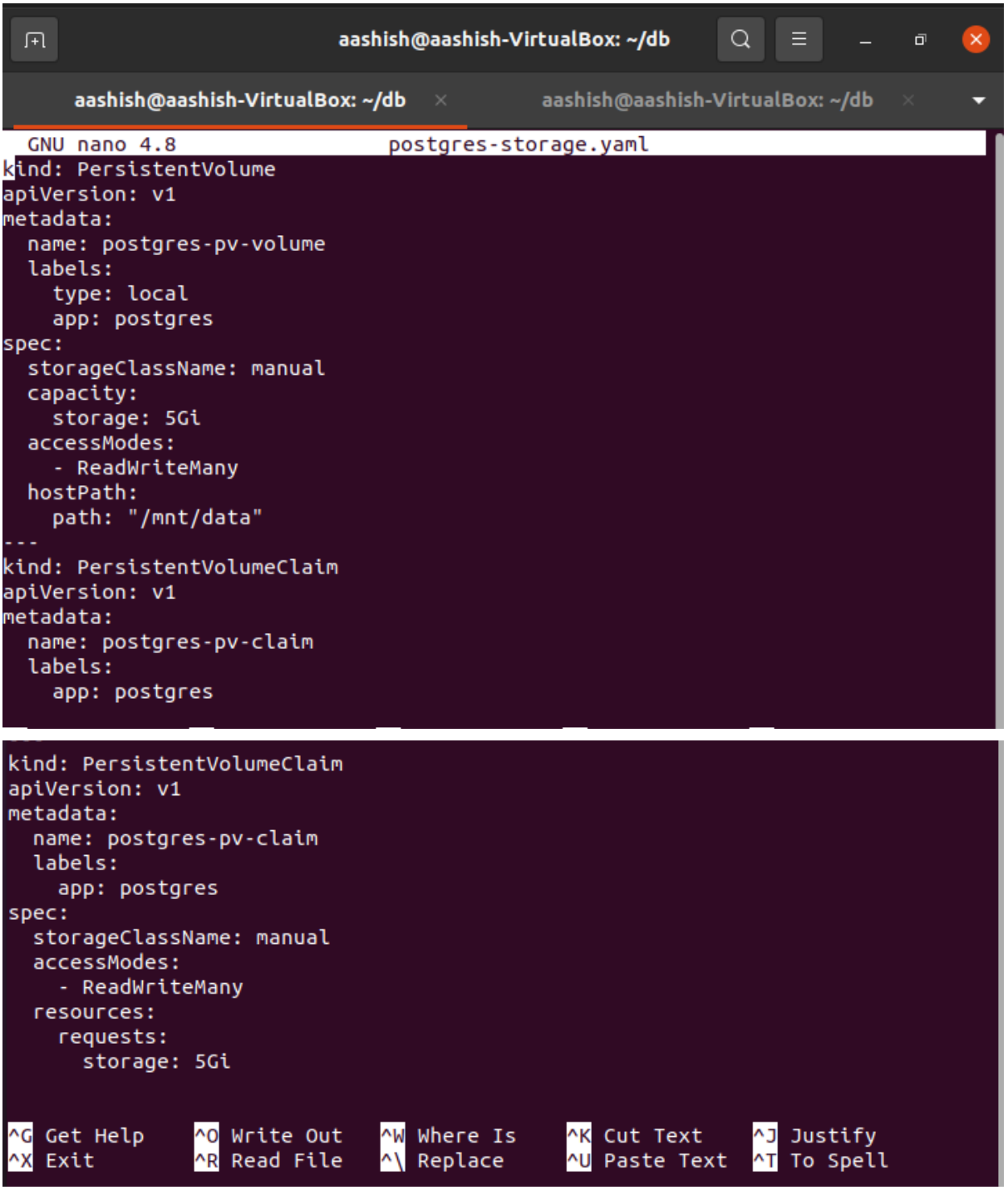



We save the file and exit. Then apply the resource with kubectl:

- **sudo kubectl apply -f postgres-configmap.yaml**

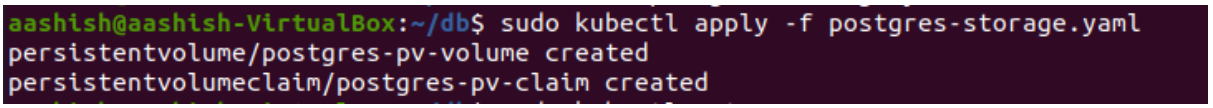



Next, we create a YAML file for storage configuration named postgres-storage.yaml using a text editor. Here, I have placed both PV and PVC configurations in one file but we can do it separately as well. The postgres-storage.yaml file is given below;




We save the file and exit. Then, apply the resources as follows;

- **sudo kubectl apply -f postgres-storage.yaml**



To check the PVC connection with PV, we use following command;

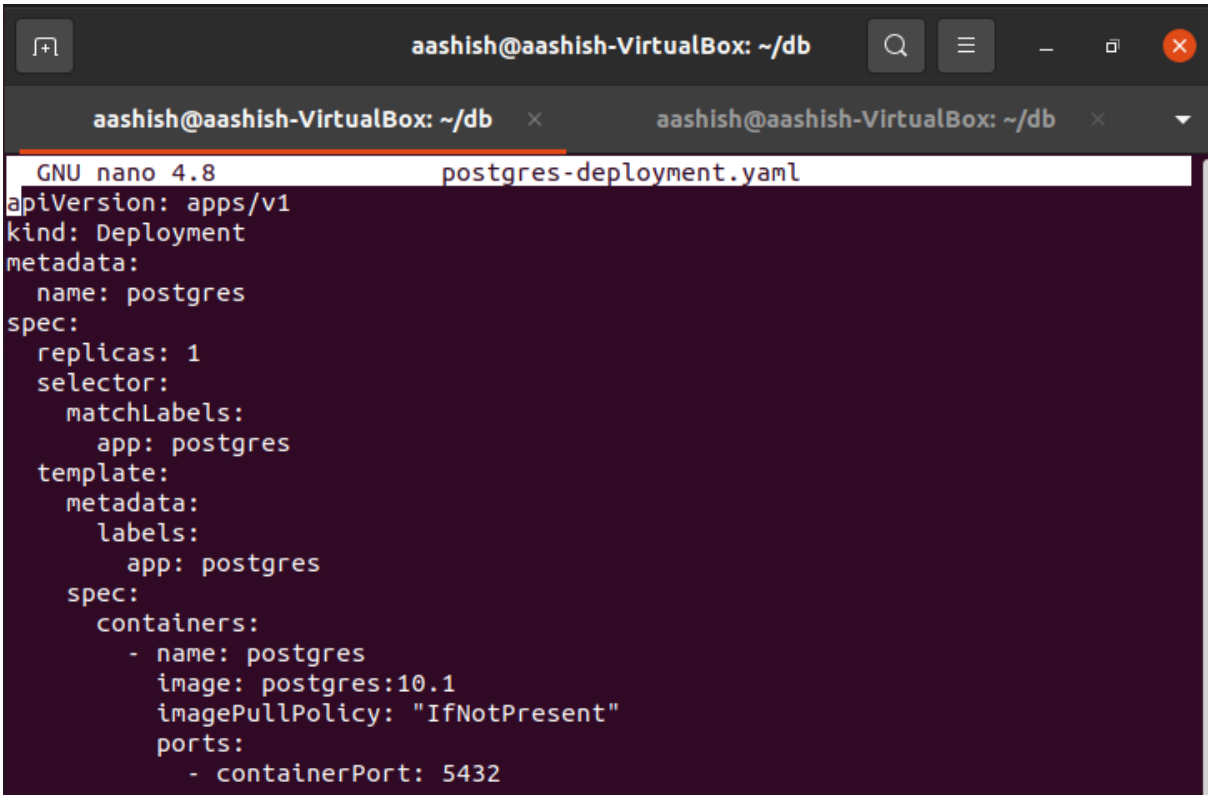
- **sudo kubectl get pvc**

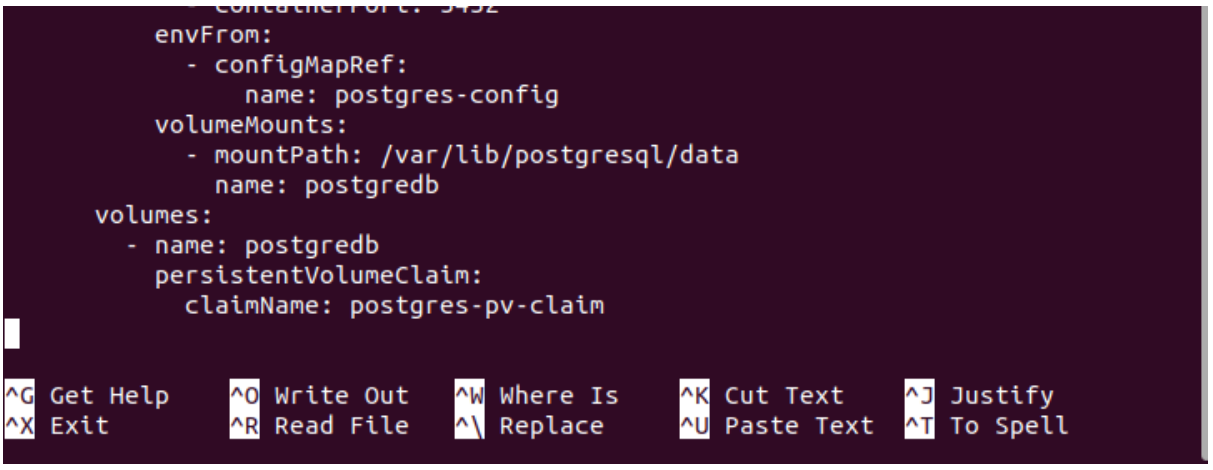
```
aashish@aashish-VirtualBox:~/db$ sudo kubectl get pvc
NAME                STATUS   VOLUME                CAPACITY   ACCESS MODES   STO
RAGECLASS   AGE
postgres-pv-claim   Bound    postgres-pv-volume    5Gi        RWX            man
ual         14m
```

The status of the PVC is Bound, and the PVC is ready to be used in the PostgreSQL deployment.

Now, we create a deployment YAML file named postgres-deployment.yaml as follows;

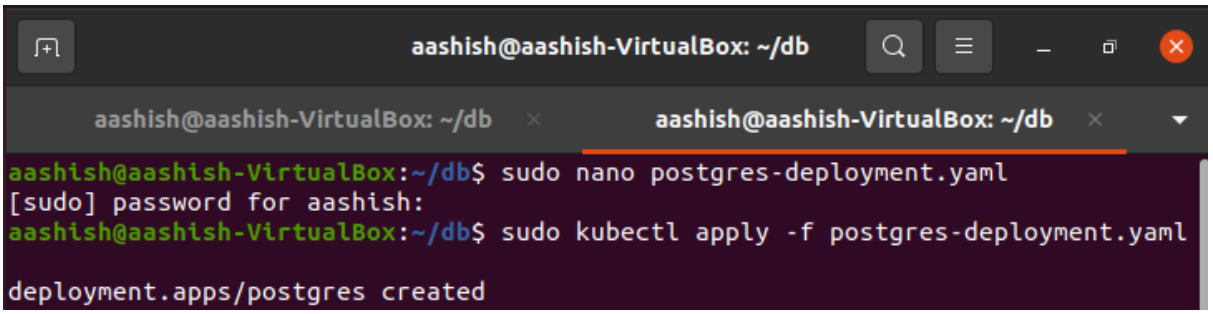
- **sudo nano postgres-deployment.yaml**



We save the file and exit. Then apply the deployment as follows;

- **sudo kubectl apply -f postgres-deployment.yaml**




Hence, the Postgres database using PV and PVC has been deployed successfully.