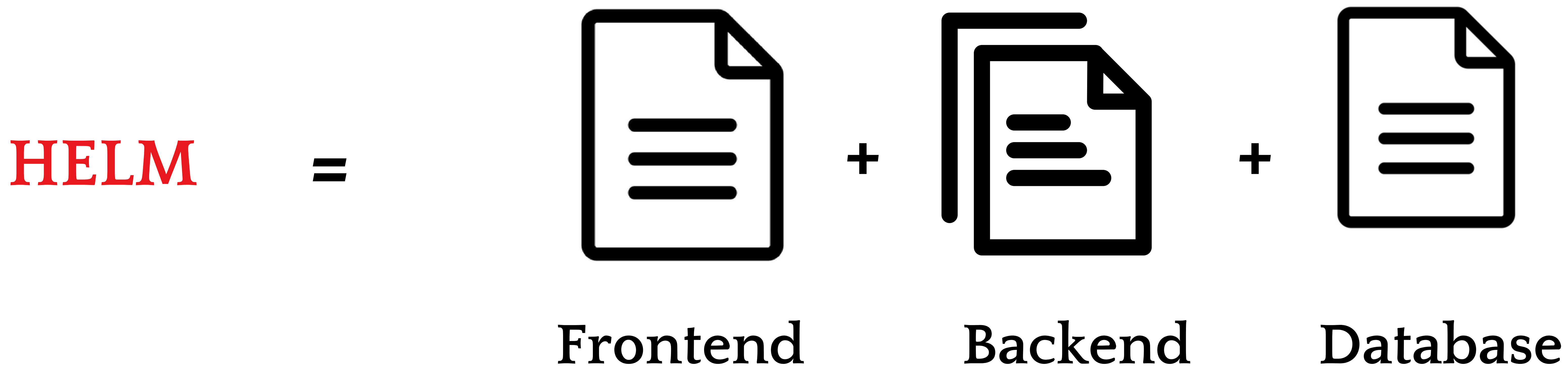
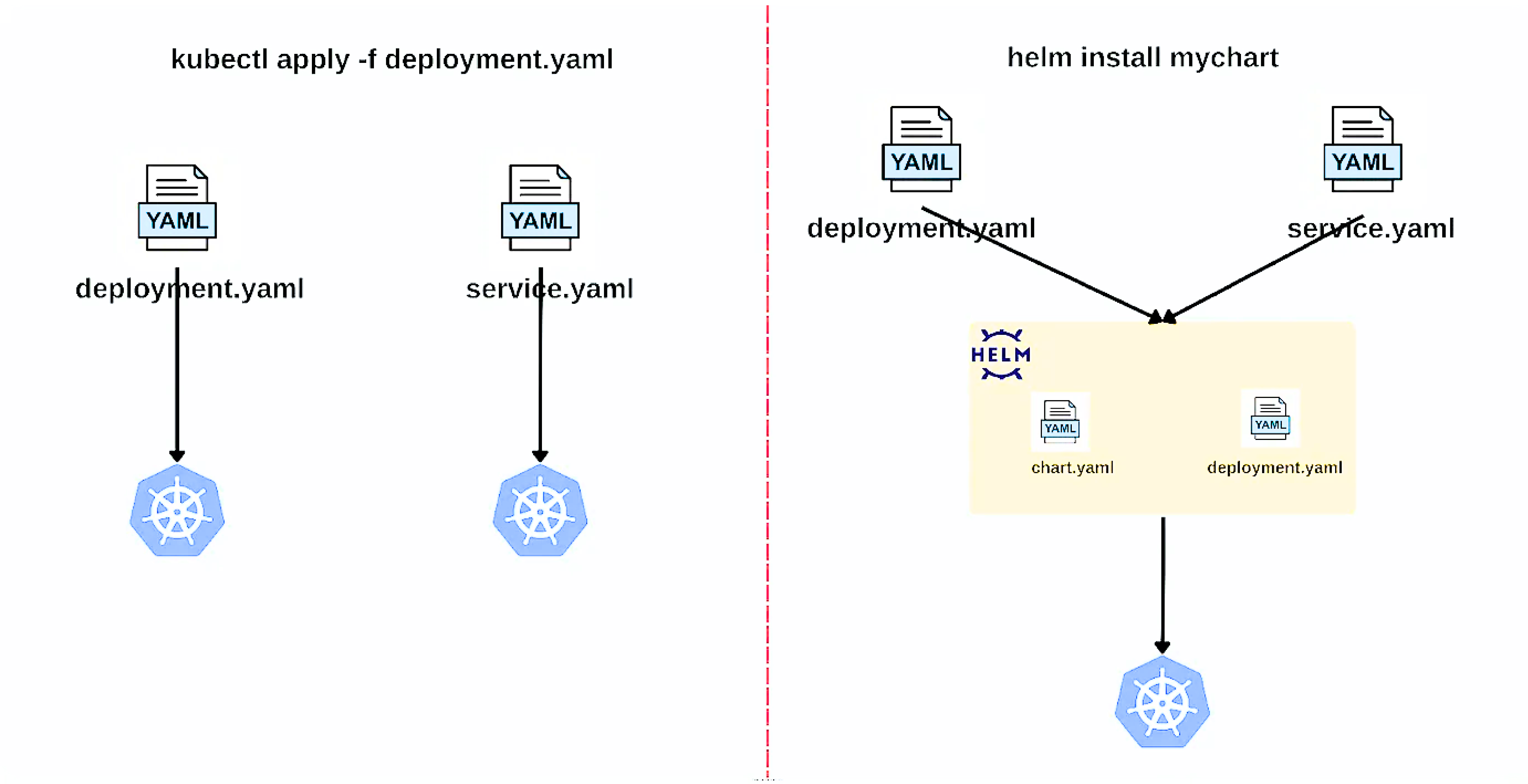
HELM

Helm is a Kubernetes package manager in which the multiple numbers of YAML files such as the backend, and frontend come under one roof(helm) and deploy using helm.




kubectl apply -f deployment.yaml
YAML
deployment.yaml
YAML
service.yaml
helm install mychart
YAML
deployment.yaml
YAML
service.yaml
HELM
YAML
chart.yaml
YAML
deployment.yaml

- Lets say if you want to deploy an application called paytm. It contains multiple services. So we need to deploy each service, for that we have to write 2 manifest(deployment & service) files for each service.
- At the end we can see multiple YAML files which makes more confusion and messy.
- Instead of maintaining those multiple YAML files we can do this by implementing HELM. It will deploy our entire application.
- HELM is a package manager for kubernetes which makes deployment simple.

Key Concepts of HEML:

- Charts: In Helm, a chart is a collection of pre-configured Kubernetes resources that describe a related set of services. Charts can include deployment configurations, service definitions, ingress rules, and more. Think of a chart as a package that encapsulates everything needed to run a specific application.
- Templates: Helm uses Go templating to dynamically generate Kubernetes manifest files from chart templates. This means you can parameterize and customize your resources based on user-defined values or other inputs. This templating mechanism allows you to create reusable and flexible configurations.

- Values: Values are customizable parameters that can be injected into chart templates during deployment. You can define default values in a values.yaml file and override them when installing or upgrading a chart. This allows you to configure your application differently for various environments or use cases.
- Repositories: Helm repositories are collections of packaged charts that users can install. The Helm CLI can search, fetch, and install charts from these repositories. You can also create your own private repository to store custom charts.

Advatnages of HEML:

It will save the time

Helps to automate the application deployment

Better scalability

Perform rollback at anytime

Increase the speed of deployment.

Install HEML:

- curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
- chmod 700 get_helm.sh
- ./get_helm.sh
- helm version

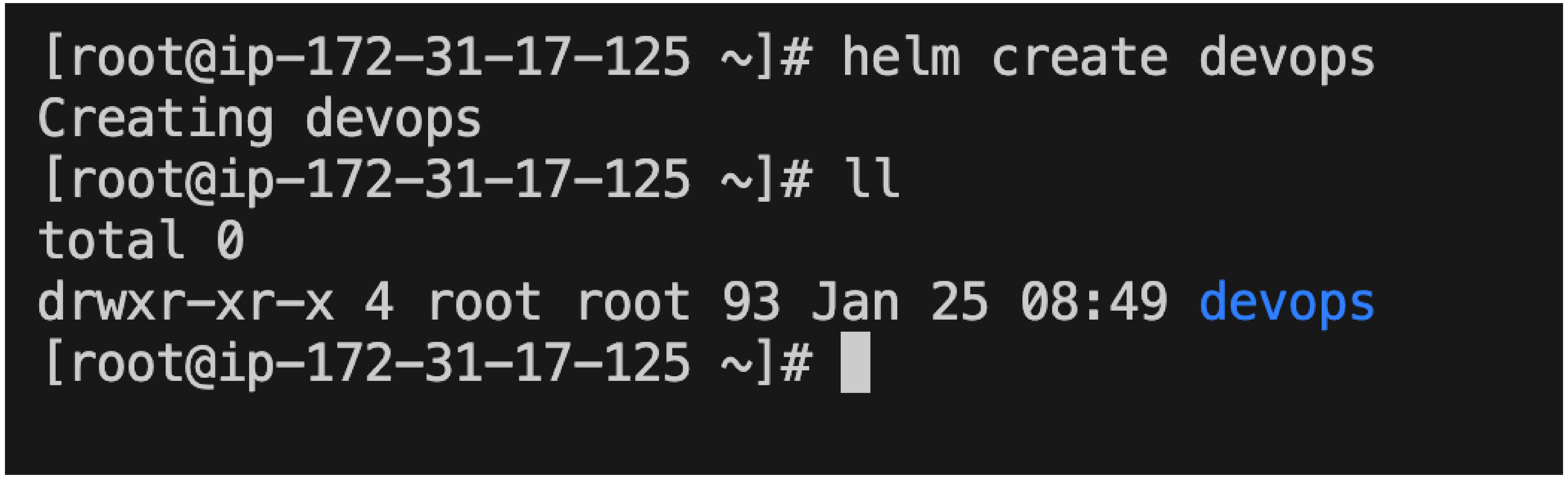
HELM commands:

- To list all helm repos : helm repo list
- To download any repo : helm repo add repo_name url (**helm repo add myhelm https://charts.helm.sh/stable**)
- To remove a repo from helm : helm repo remove repo-names

Deploy a sample Nginx page using HELM:

1. Create a helm chart : helm create devops

   Now we can see devops folder will gets created.



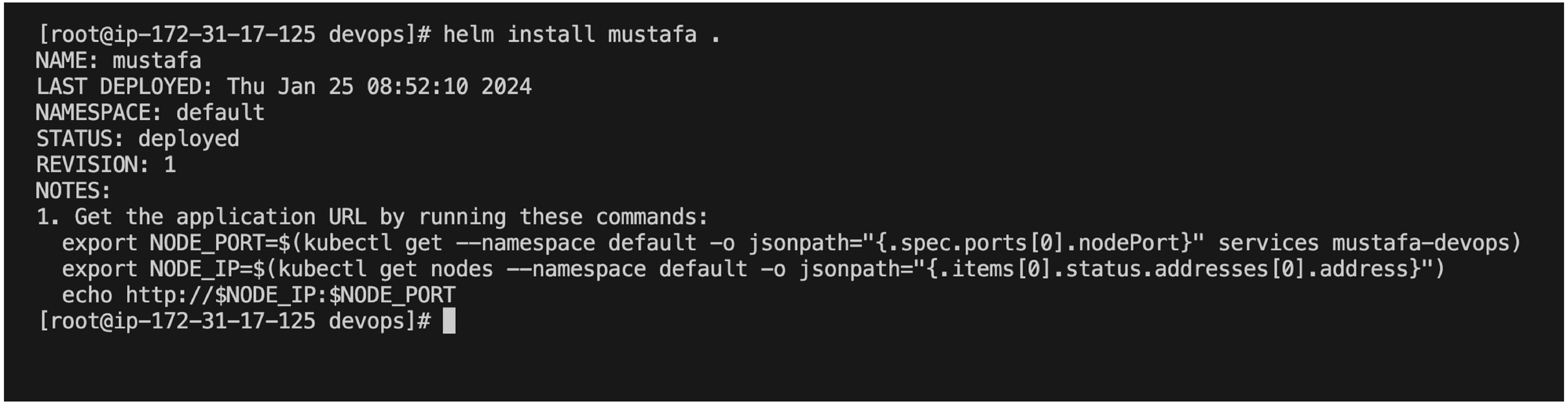
```
[root@ip-172-31-17-125 ~]# helm create devops
Creating devops
[root@ip-172-31-17-125 ~]# ll
total 0
drwxr-xr-x 4 root root 93 Jan 25 08:49 devops
[root@ip-172-31-17-125 ~]#
```

2. Enter into the folder and open **values.yml** file
3. Change the service type from **ClusterIP to NodePort** on line 43

4. Now install helm : helm install mustafa .

here dot(.) represents that pwd where our Chart.yaml present

mustafa is release name



```
[root@ip-172-31-17-125 devops]# helm install mustafa .
NAME: mustafa
LAST DEPLOYED: Thu Jan 25 08:52:10 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0].nodePort}" services mustafa-devops)
  export NODE_IP=$(kubectl get nodes --namespace default -o jsonpath="{.items[0].status.addresses[0].address}")
  echo http://$NODE_IP:$NODE_PORT
[root@ip-172-31-17-125 devops]# 
```

5. Now we can see pods, deployments, services will be created automatically.

Now we can access those nginx page using slaveip:node-port

6. To get helm release list : helm list (or) helm ls

6. Now lets scale up the replicas in values.yml (change value from 1 to 4)
7. To update those changes we will use : helm upgrade mustafa .
8. Now lets see the pods count, it will gets increased from 1 to 4.
9. Now we can see the release using the command helm list then revision(version) will gets increased from 1 to 2
10. To rollback to prev revision : helm rollback mustafa 1
The above command will rollback to 1st version
Once we perform the above command we can see the pods count will be scaledown from 4 to 1 again.

General commands about HELM:

- To uninstall helm : helm uninstall mustafa
- If you want to dry-run the chart before installation : helm install devops --debug --dry-run .
- Validate all the YAML files in our HELM charts : helm list .
- To get template of our helm : helm template .
- To see the history of a release : helm history mustafa