HelmFile

Why use Helmfile?

Helm is a great tool for templating and sharing Kubernetes manifests for your applications. However it can become quite challenging to install larger multi-tier applications or groups of applications across multiple Kubernetes clusters.

Helmfile addresses this issue and more by providing a fairly simple but very powerful declarative specification for deploying Helm charts across many environments.

Helmfile is a declarative spec for deploying helm charts

Helmfile is another wrapper working on top of Helm Chart. Just like Helm Chart, Helmfile also uses the YAML for writing the configurations.

Benefits of Helmfile:

1. You can bundle several Helm Charts into a Single Helmfile to manage your kubernetes ecosystem
2. Helmfile maintains the state file like the terraform. It can help you to identify the differences between the new changes which you want to apply against the existing running deployment inside kubernetes cluster

Installing the helmfile:

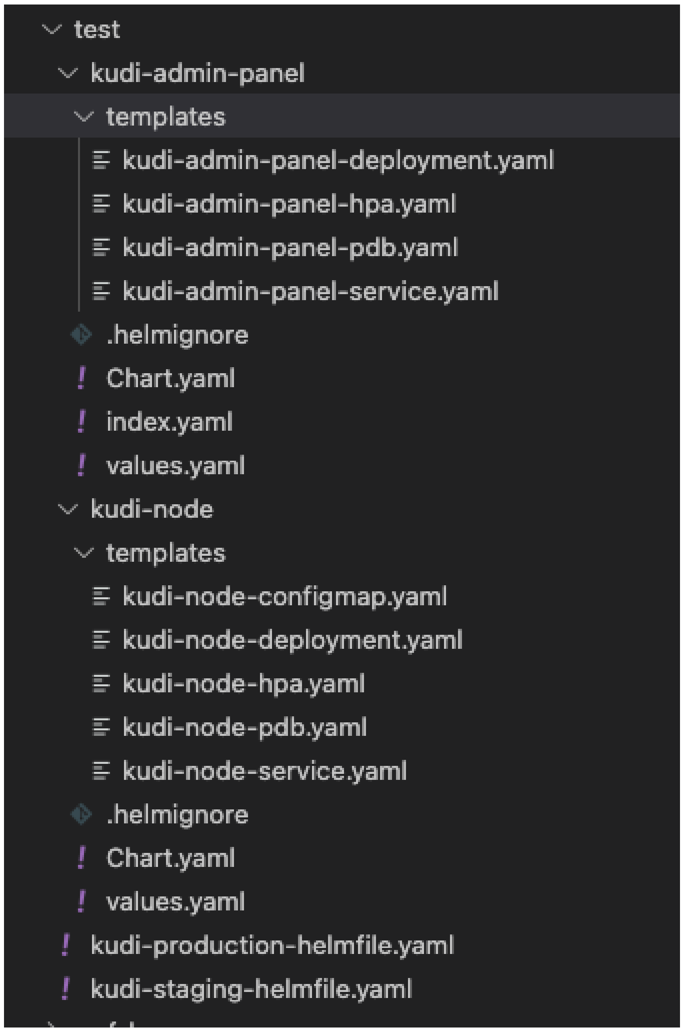
- Windows (using scoop): scoop install helmfile
- macOS (using homebrew): brew install helmfile
- In Linux:
  https://github.com/roboll/helmfile/releases

  ```
  wget {{ bin_url }} -O helmfile
  chmod +x helmfile
  mv helmfile /usr/local/bin
  ```

Plugins required for helmfile:

- Helm Diff plugin:
  `helm plugin install https://github.com/databus23/helm-diff`

Helmfile folder structure:



```
test
  kudi-admin-panel
    templates
      kudi-admin-panel-deployment.yaml
      kudi-admin-panel-hpa.yaml
      kudi-admin-panel-pdb.yaml
      kudi-admin-panel-service.yaml
    .helmignore
    Chart.yaml
    index.yaml
    values.yaml
  kudi-node
    templates
      kudi-node-configmap.yaml
      kudi-node-deployment.yaml
      kudi-node-hpa.yaml
      kudi-node-pdb.yaml
      kudi-node-service.yaml
    .helmignore
    Chart.yaml
    values.yaml
  kudi-production-helmfile.yaml
  kudi-staging-helmfile.yaml
```

Sample helmfile for using helm chart deployment:

helmfile.yaml

```yaml
releases:
  - name: prometheus-operator
    namespace: monitoring
    chart: prometheus-community/kube-prometheus-stack
```

For using the existing helm charts in helmfile:

```
repositories:
 - name: stable
   url: https://charts.helm.sh/stable

releases:
 - name: kudi-admin-panel-staging
   chart: ./kudi-admin-panel-staging/
   values:
       - ./kudi-admin-panel-staging/values.yaml
   installed: true
```

Here, repositories are the repositories that are going to add to the helm chart.
releases are the release name of the helm deployment.

Installed is the state of the helm chart,
If (installed: true) then it will install the helm chart
If (installed: false) then it will uninstall the helm chart

Helmfile commands:

- helmfile repos
  It will automatically fetch and update the repositories from the state file.
- helmfile apply
  To apply the changes detected in the state file.
- helmfile status
  To check the states of the helm charts.
- helmfile test
  To test the configuration in the helmfile.
- helmfile destroy
  To destroy the entire infrastructure created using the helmfile.

By default, the name of the helmfile is helmfile.yaml,
When we apply the helm apply it will automatically detect the helmfile.yaml and applies it.
In case, we are having the multiple helmfiles for separate environments, we can manually pass the file name to apply.

- helmfile apply -f helm-file-1.yaml

By default, the helm tries to create the new namespace with the mentioned namespace; this can cause errors.
To avoid creating namespace we need to mention explicitly,

```
helmDefaults:
 wait: true
 timeout: 600
 force: false
 createNamespace: false
```

For more references on advanced configurations of helmfile:

https://lyz-code.github.io/blue-book/devops/helmfile/#how-to-deploy-a-new-chart
https://github.com/helmfile/helmfile
https://medium.com/swlh/how-to-declaratively-run-helm-charts-using-helmfile-ac78572e6088
https://github.com/cloudposse/helmfiles
https://jhooq.com/helmfile-manage-helmchart/
https://github.com/roboll/helmfile