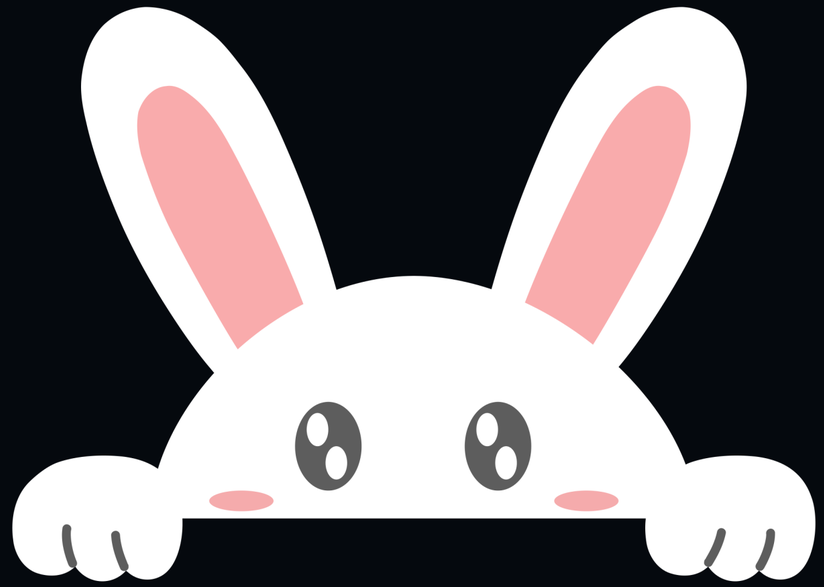
HELM BEHIND SCENE

- Check local & remote repository and load it with its dependencies
- Check if there is any custom values.yaml or value set
- Merging the custom values to default and generate the yaml
- Validate all yaml is proper
- Send to Kube



HELM DRY RUN

- release information with the pending execution

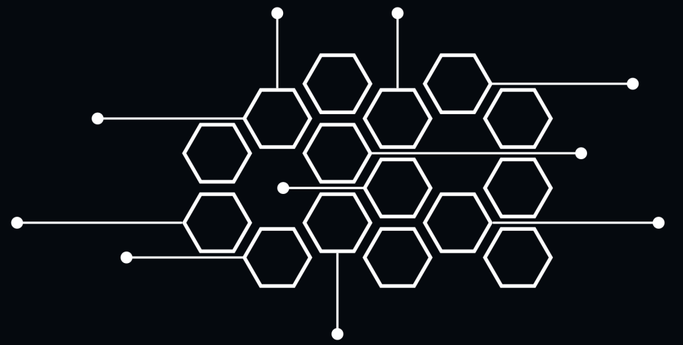
```
# dry run
helm install mydb bitnami/mysql -f my-values.yaml --dry-run

```

HELM TEMPLATE

- No release information
- No need to access to Kubernetes server
- show **installation(not upgrade)** yaml

```
1  # template
2  helm template mydb bitnami/mysql -f my-values.yaml
```



HELM GET

```
# get notes information
helm get notes mydb

# get customer values(not default value)
helm get values mydb

# get customer values(including default value)
helm get values mydb --all

# get revision value
helm get values mydb --revision 1

# get manifest of current release
helm get manifest mydb
```

HELM INSTALL

```
# install or upgrade
helm upgrade --install my-wordpress bitnami/wordpress

# within custom namespace
helm install mydb bitnami/mysql --namespace ns1 --create-namespace

# with random name
helm install bitnami/mysql --generate-name

# mark as failure after waiting time(default 5min)
helm install bitnami/mysql --generate-name --wait --timeout 2m30s

# atomic(rollback to prev successful release)
helm install bitnami/mysql --generate-name --atomic
```

HELM UPGRADE

```
# upgrade
helm upgrade my-wordpress bitnami/wordpress -f my-values.yaml

# upgrade forcefully(will have the downtime)
helm upgrade my-wordpress bitnami/wordpress -f my-values.yaml --force

# cleanup on failed
helm upgrade my-wordpress bitnami/wordpress -f my-values.yaml --cleanup-on-failure
```