Helm KodeKloud

Sunday, 11 June 2023 10:06 PM

Section	Description
What is Helm?	Helm - package manager for kubernetes Why helm might be beneficial? • For an application, we might need multiple YAML files e.g. deployment, secrets, etc • We need custom parameters for the values in the fields of each yaml file • We might also need versioning • We can upgrade our application with a single command ○ helm upgrade wordpress • Rollback easier ○ helm rollback wordpress Customization • If we were to for example, deploy prometheus • we can include settings for ldap, size of PV, admin password, etc

Installation	Install via
	• snap install helm --classic
	• the classic parameter allows the access to the kubeconfig in our home directory so helm can easily access the cluster
	
	Can configure your helm parameters via the environment variables
	
	`controlplane ~ ✖ helm help`
	`The Kubernetes package manager`
	
	`Common actions for Helm:`
	
	`- helm search: search for charts`
	`- helm pull: download a chart to your local directory to view`
	`- helm install: upload the chart to Kubernetes`
	`- helm list: list releases of charts`
	
	`Environment variables:`
	
	`| Name | Description |`
	`| $HELM_CACHE_HOME | set an alternative location for storing cached files. |`
	`| $HELM_CONFIG_HOME | set an alternative location for storing Helm configuration. |`
	`| $HELM_DATA_HOME | set an alternative location for storing Helm data. |`
	`| $HELM_DEBUG | indicate whether or not Helm is running in Debug mode |`
	`| $HELM_DRIVER | set the backend storage driver. Values are: configmap, secret, memory, sql. |`
	`| $HELM_DRIVER_SQL_CONNECTION_STRING | set the connection string the SQL storage driver should use. |`
	`| $HELM_MAX_HISTORY | set the maximum number of helm release history. |`
	`| $HELM_NAMESPACE | set the namespace used for the helm operations. |`
	`| $HELM_NO_PLUGINS | disable plugins. Set HELM_NO_PLUGINS=1 to disable plugins. |`
	`| $HELM_PLUGINS | set the path to the plugins directory |`
	`| $HELM_REGISTRY_CONFIG | set the path to the registry config file. |`
	`| $HELM_REPOSITORY_CACHE | set the path to the repository cache directory |`
	`| $HELM_REPOSITORY_CONFIG | set the path to the repositories file. |`
	`| $KUBECONFIG | set an alternative Kubernetes configuration file (default "~/.kube/config") |`
	`| $HELM_KUBEAPISERVER | set the Kubernetes API Server Endpoint for authentication |`
	`| $HELM_KUBECAFILE | set the Kubernetes certificate authority file. |`
	`| $HELM_KUBEASGROUPS | set the Groups to use for impersonation using a comma-separated list. |`
	`| $HELM_KUBEASUSER | set the Username to impersonate for the operation. |`
	`| $HELM_KUBECONTEXT | set the name of the kubeconfig context. |`
	`| $HELM_KUBETOKEN | set the Bearer KubeToken used for authentication. |`
	`| $HELM_KUBEINSECURE_SKIP_TLS_VERIFY | indicate if the Kubernetes API server's certificate validation should be skipped (insecure) |`
	`| $HELM_KUBETLS_SERVER_NAME | set the server name used to validate the Kubernetes API server certificate |`
	`| $HELM_BURST_LIMIT | set the default burst limit in the case the server contains many CRDs (default 100, -1 to disable)|`
	
	`Helm stores cache, configuration, and data based on the following configuration order:`
	
	`- If a HELM_*_HOME environment variable is set, it will be used`
	`- Otherwise, on systems supporting the XDG base directory specification, the XDG variables will be used`
	`- When no other location is set a default location will be used based on the operating system`
	
	`By default, the default directories depend on the Operating System. The defaults are listed below:`
	
	`| Operating System | Cache Path | Configuration Path | Data Path |`
	`| Linux | $HOME/.cache/helm | $HOME/.config/helm | $HOME/.local/share/helm |`
	`| macOS | $HOME/Library/Caches/helm | $HOME/Library/Preferences/helm | $HOME/Library/helm |`
	`| Windows | %TEMP%\helm | %APPDATA%\helm | %APPDATA%\helm |`
	
	`Usage:`
	`helm [command]`

Helm Components	Charts
	• collection of files
	• contain all instructions that helm need sto be able to create the collection of objects in the cluster
	
	service.yaml
	`apiVersion: v1`
	`kind: Service`
	`metadata:`
	`name: hello-world`
	`spec:`
	`type: NodePort`
	`ports:`
	`- port: 80`
	`targetPort: http`
	`protocol: TCP`
	`name: http`
	`selector:`
	`app: hello-world`
	
	deployment.yaml
	`apiVersion: apps/v1`
	`kind: Deployment`
	`metadata:`
	`name: hello-world`
	`spec:`
	`replicas: {{ .Values.replicaCount }}`
	`selector:`
	`matchLabels:`
	`app: hello-world`
	`template:`
	`metadata:`
	`labels:`
	`app: hello-world`
	`spec:`
	`containers:`
	`- name: nginx`
	`image: "{{ .Values.image.repository }}"`
	`ports:`
	`- name: http`
	`containerPort: 80`
	`protocol: TCP`
	
	values.yaml
	`replicaCount: 1`
	
	`image:`
	`repository: nginx`
	
	The {{ .Values.image }} is called templating - normally the templates can be used just as it is
	• just need to update the values.yaml file
	
	The values.yaml file is like the input files for the helm chart

```
# helm install [release-name][chart-name]

$ helm install my-site bitnami/wordpress


# helm install bitnami/wordpress


$ helm install my-SECOND-site bitnami/wordpress
```

Need to have a release name for a chart
- this is because we can install the same charts
- e.g. install a wordpress for internal/external

Helm repositories
- thousand of charts are available on repositories around the world
- e.g. artifacthub.io

Helm Charts

```yaml
apiVersion: v2
appVersion: 5.8.1
version: 12.1.27
name: wordpress
description: Web publishing platform for building blogs and websites.
type: application
dependencies:
  - condition: mariadb.enabled
    name: mariadb
    repository: https://charts.bitnami.com/bitnami
    version: 9.x.x
    <code hidden>
keywords:
  - application
  - blog
  - wordpress
maintainers:
  - email: containers@bitnami.com
    name: Bitnami
home: https://github.com/bitnami/charts/tree/master/bitnami/wordpress
icon: https://bitnami.com/assets/stacks/wordpress/img/wordpress-stack-220x234.png
```

apiVersion: v1 --> if you use helm2, but if helm3, use v2
appVersion: --> version of the application thats inside of this chart - version of wordpress that is being deployed (for informational purposes only)
version: version of the chart itself
type: 2 types of charts - application/libraray
- application is default
- library provides utility that helps in building charts

dependencies
- need to install mariadb before this
- no need another chart for mariadb, can use this to install mariadb

keywords
- good to put for searching stuff in gitlab

rest is informational

Helm Chart Structure

```
hello-world-chart
  templates      # Templates directory
  values.yaml    # Configurable values
  Chart.yaml     # Chart information
  LICENSE        # Chart License
  README.md      # Readme file
  charts         # Dependency Charts
```

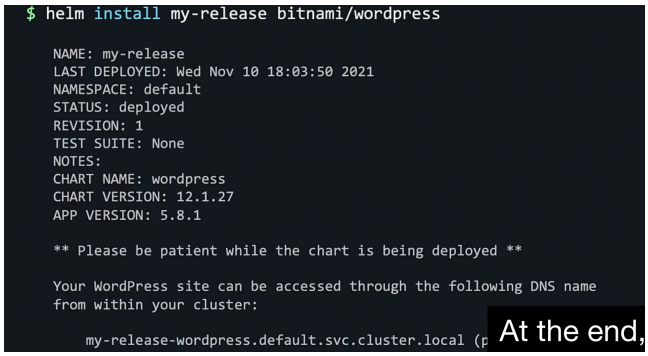
Working with Helm: basics

helm --help
- quick way to do stuff

When we need to launch a wordpress website in Kubernetes
- Searching from chart
  - We know charts are on artifacthub.io, and we can check from that website oR
  - helm search hub wordpress
    - the hub refers to artifacthub.io (default)
  - If want to add repo, need ot use the repo function
    - helm repo add bitanami http://charts.bitnami.com/bitnami
    - helm install my-release bitnami/wordpress

```
$ helm repo add bitnami https://charts.bitnami.com/bitnami

   "bitnami" has been added to your repositories
```



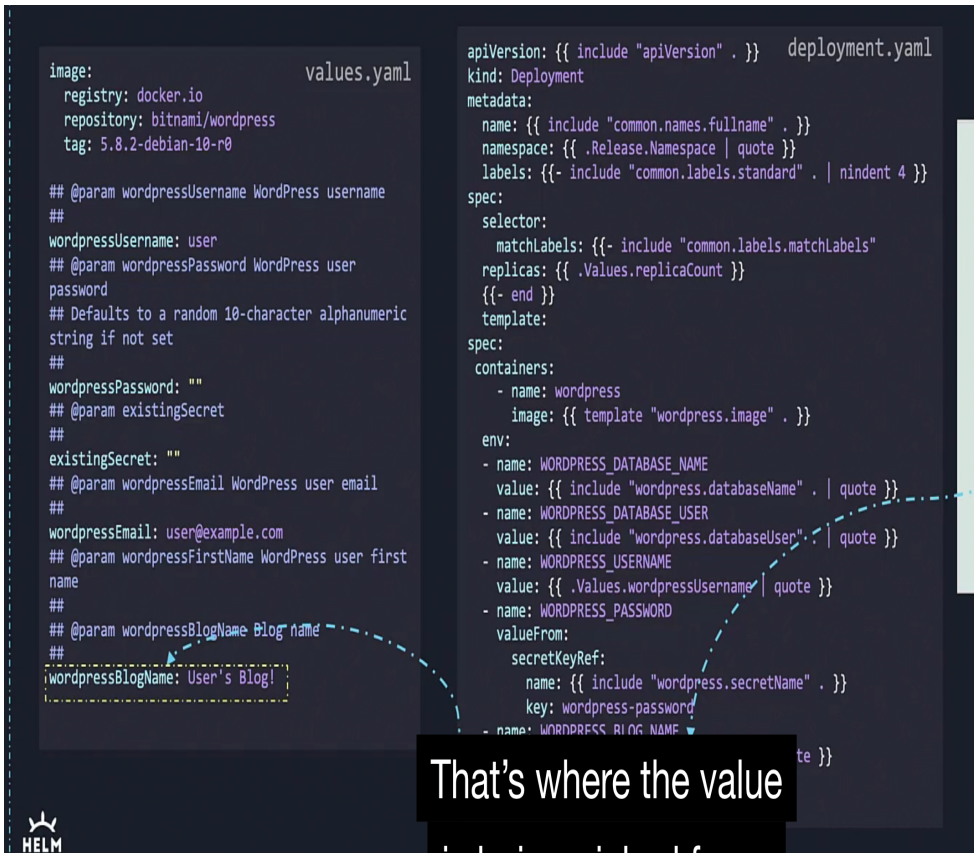
Once a chart is deployed, it is deployed as a release
- `helm list`
- when we want to delete, we don't have to remove them from the cluster one by one
  - we can just *helm uninstall my-release*

Adding a repo to helm
- helm repo add
- helm repo list
- helm repo update (similar to yum update) - to get latest data

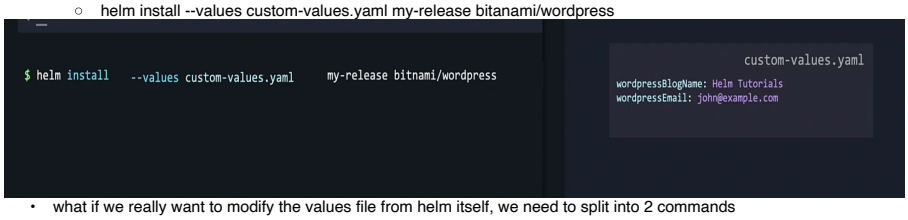
Cusotmizing chart parameters

In the above example, we installed wordpress with the default value.

We may not want the word press site to have the default name
- the wordpress deployment file contains templating values for the environment variables which will affect the wordpress name



But if we do a "helm install my-release bitnami/wordpress", we got no chance to set the values in values.yaml
- we need to use the --set value e.g.
- helm install --set wordpressBlogName="Helm Tuts" --set wordpressEmail="xxx" my-release bitnami/wordpress OR
- create a custom values.yaml file
  - helm install --values custom-values.yaml my-release bitanami/wordpress



- what if we really want to modify the values file from helm itself, we need to split into 2 commands
  - helm pull --untar bitnami/wordpress
    - this creates a directory called wordpress, with all the values file in the directory
    - we can then open and edit the values.yaml
  - helm install my-release ./wordpress (the directory)
    - this is done after editing the values.yaml file

Lifecycle management with helm

Upgrade the deployment with helm
- helm upgrade nginx-release bitnami/nginx

We can see the current revision number incremented to 2

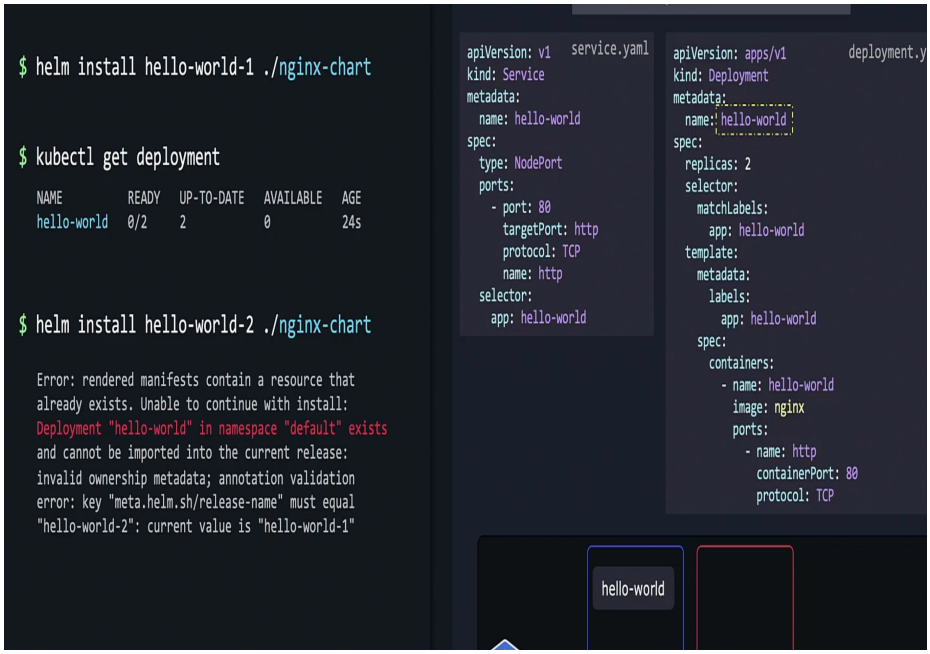
helm list
helm history nginx-release --> can see a lot of useful releases
helm rollback nginx-release <revision number>

Writing a Helm Chart

Need to create a directory structure first

Or perform a command

- "helm create weka-mon-exporter"
- it will generate some files with sample values
- edit and place your service and deployment files in templates/

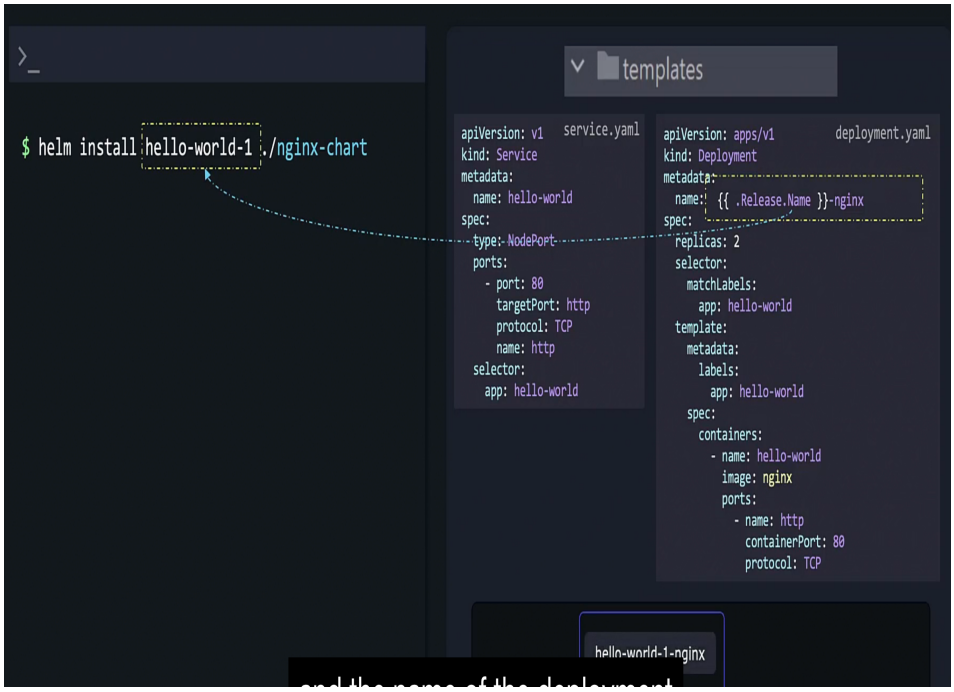


Cannot have the same deployment with the same name

From above, the templates/deployment.yaml contains a static name called "Hello World"

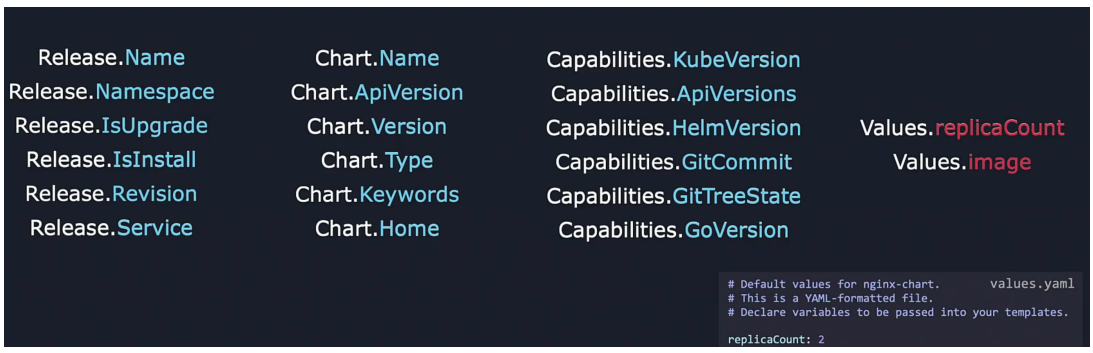
We need to template this value,

It should be

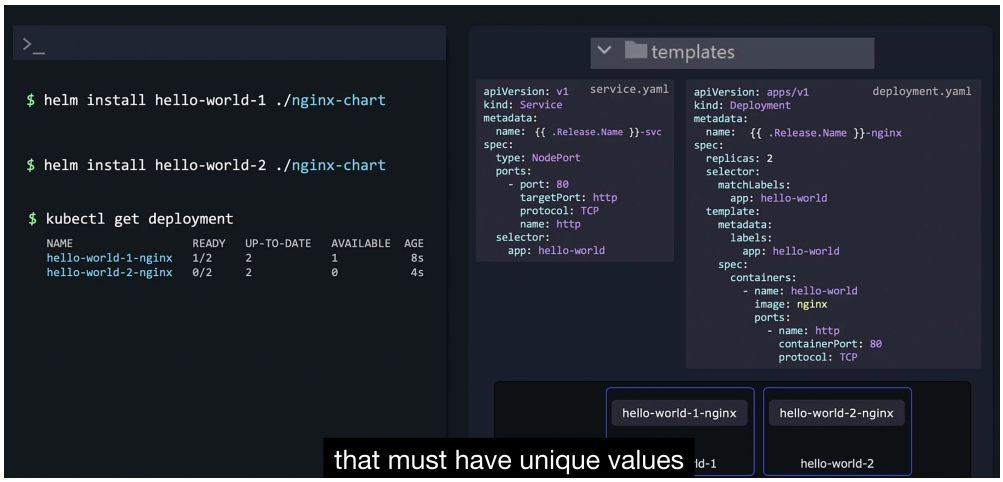


The values that you can specify (if you use helm)

- note the capitalization for the Release/Chart/Capabilities part

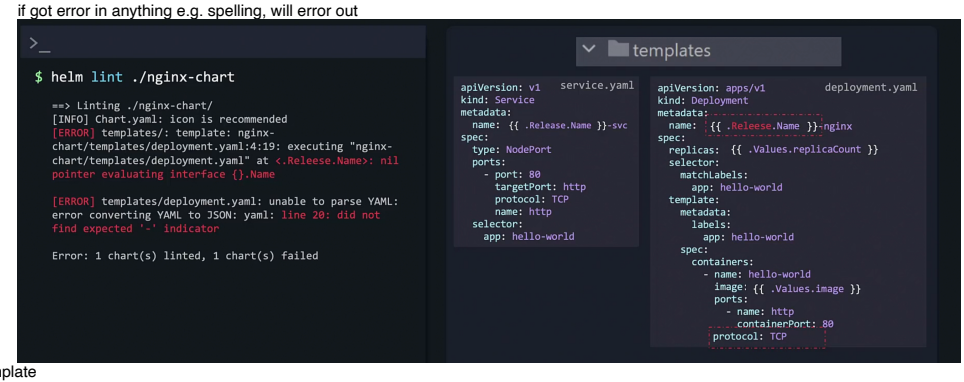


Heres how we can install 2 releases (template the name of the deployment/service)



Making sure Chart is working as intended

How do we ensure the chart works well with Kubernetes / working as intended?

- Lint
  - how do we ensure chart is built properly without any formatting errors or wrong values
  - helm lint ./nginx-chart
  - if got error in anything e.g. spelling, will error out
  - 

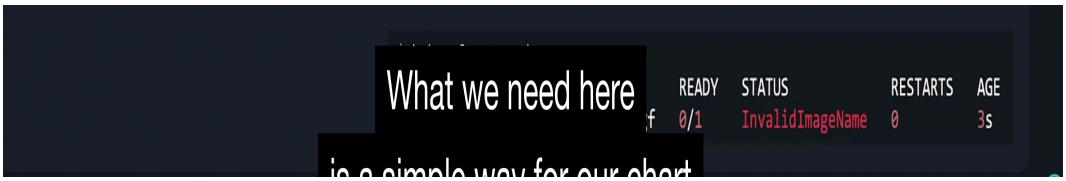
- Template
  - helm template <chart_name>
  - renders the output as per the values.yaml into the templates/
  - if got error, will error out
- Dry Run
  - the two above can help to catch many errors but cannot catch
    - typo of containers vs container
    - only kubernetes can tell us if this is wrong
  - this will pretend to install on the cluster
  - will talk to kubernetes to ensure everything is ok
  - helm install hello-world-1 ./nginx-chart --dry-run

Jerome - can you do this as a .bashrc thing for the charts

Functions

What if the values.yaml doesn't have a field set?

That will create the output file without that field



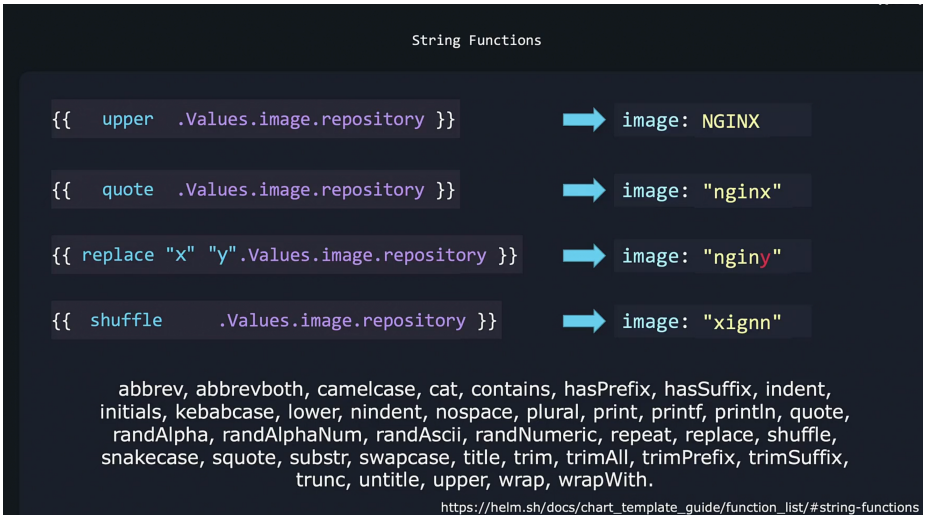
We can have a default values in case the user don't provide something in the values.yaml file

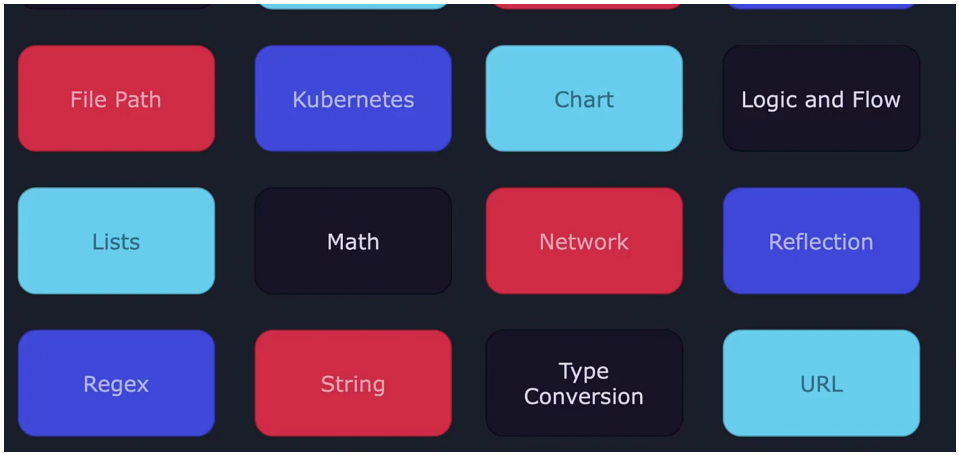
- if they provide, then it overrides
- otherwise, we can place a default value

Example

Functions that we can use

60+ Funcitons that we can use
https://helm.sh/docs/chart_template_guide/functions_and_pipelines/
https://helm.sh/docs/chart_template_guide/function_list/

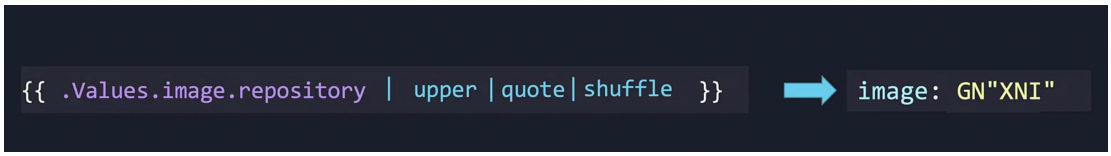




Pipelines

You can pipe functions like this
- the functions are upper/quote/shuffle

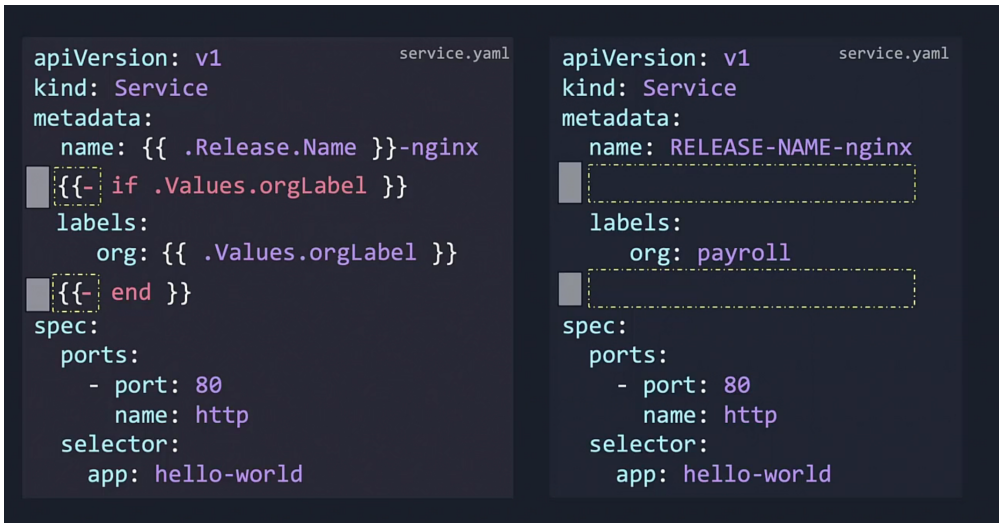
Chaining these functions with a pipe is known as a pipeline



Conditionals

If statements in the helm file

If you don't put a dash after the {{, your resulting yaml file will have the extra blank space / newline there



```
spec:
  ports:
    - port: 80
      name: http
  selector:
    app: hello-world
```

```
      name: http
  selector:
    app: hello-world
```

else if statements

```
apiVersion: v1                                          service.yaml
kind: Service
metadata:
  name: {{ .Release.Name }}-nginx
  {{- if .Values.orgLabel }}
  labels:
    org: {{ .Values.orgLabel }}
  {{- else if eq .Values.orgLabel "hr" }}
  labels:
    org: human resources
  {{- end }}
spec:
  ports:
    - port: 80
      name: http
  selector:
    app: hello-world
```

Function	Purpose
eq	equal
ne	not equal
lt	less than
le	less than or equal to
gt	greater than
ge	greater than or equal to
not	negation
empty	value is empty

This seems to be similar to terraform,

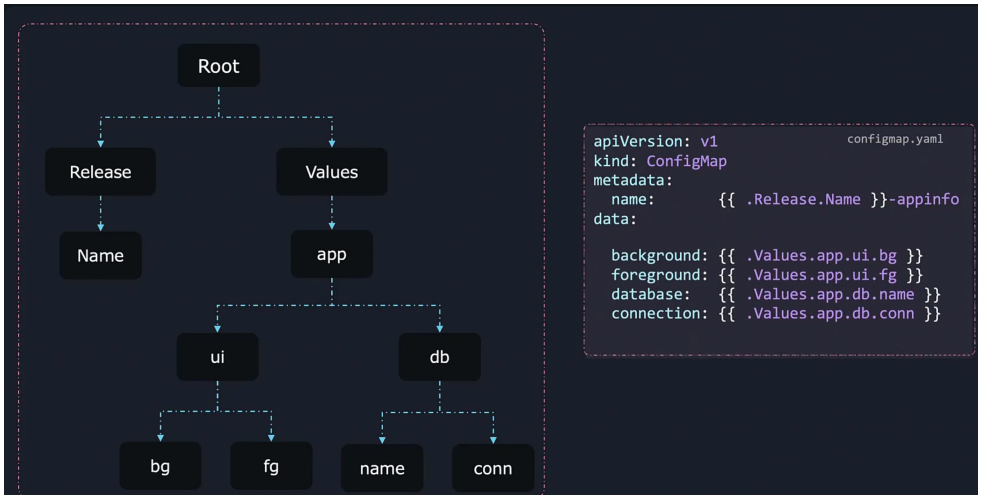
- for this can set up illumio helm enable daemonset, if "enable" then we put on all hosts

```
# Default values for nginx-chart.                       values.yaml
# This is a YAML-formatted file.


serviceAccount:
  # Specifies whether a ServiceAccount should be created
  create: true
```

```
{{- if .Values.serviceAccount.create }}                 serviceaccount.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: {{ .Release.Name }}-robot-sa
{{- else }}
```

With Blocks

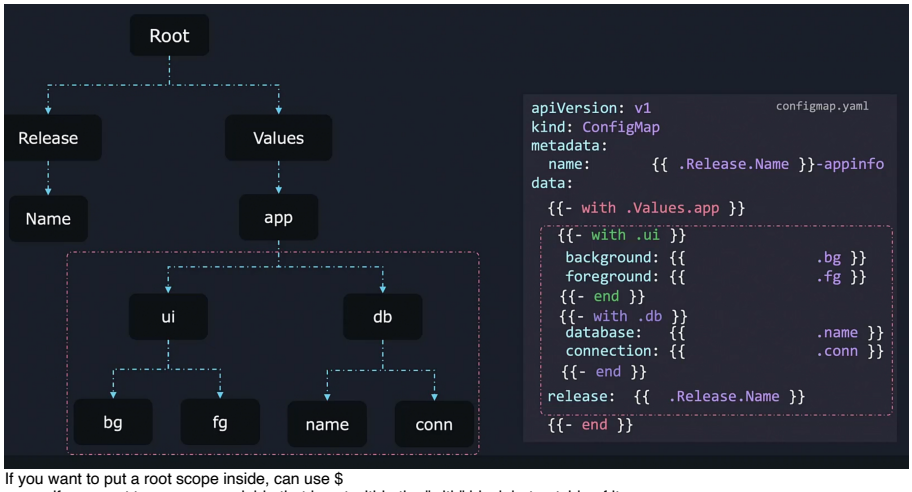



```
apiVersion: v1                                          configmap.yaml
kind: ConfigMap
metadata:
  name:       {{ .Release.Name }}-appinfo
data:

  background: {{ .Values.app.ui.bg }}
  foreground: {{ .Values.app.ui.fg }}
  database:   {{ .Values.app.db.name }}
  connection: {{ .Values.app.db.conn }}
```

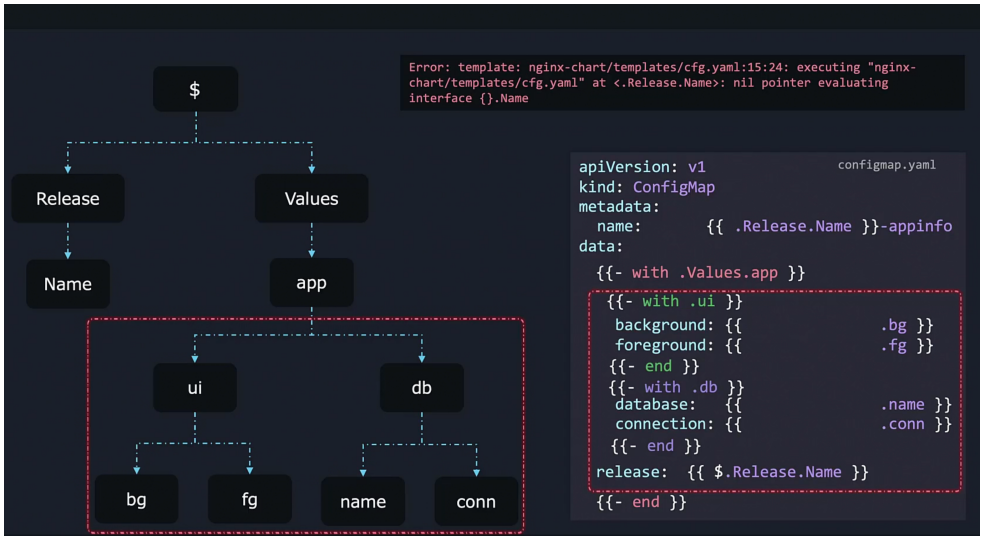
We see that there are duplications for multiple .Values.app., we can change to

```
apiVersion: v1                                          configmap.yaml
kind: ConfigMap
metadata:
  name:       {{ .Release.Name }}-appinfo
data:
  {{- with .Values.app }}
  {{- with .ui }}
    background: {{                .bg }}
    foreground: {{                .fg }}
```

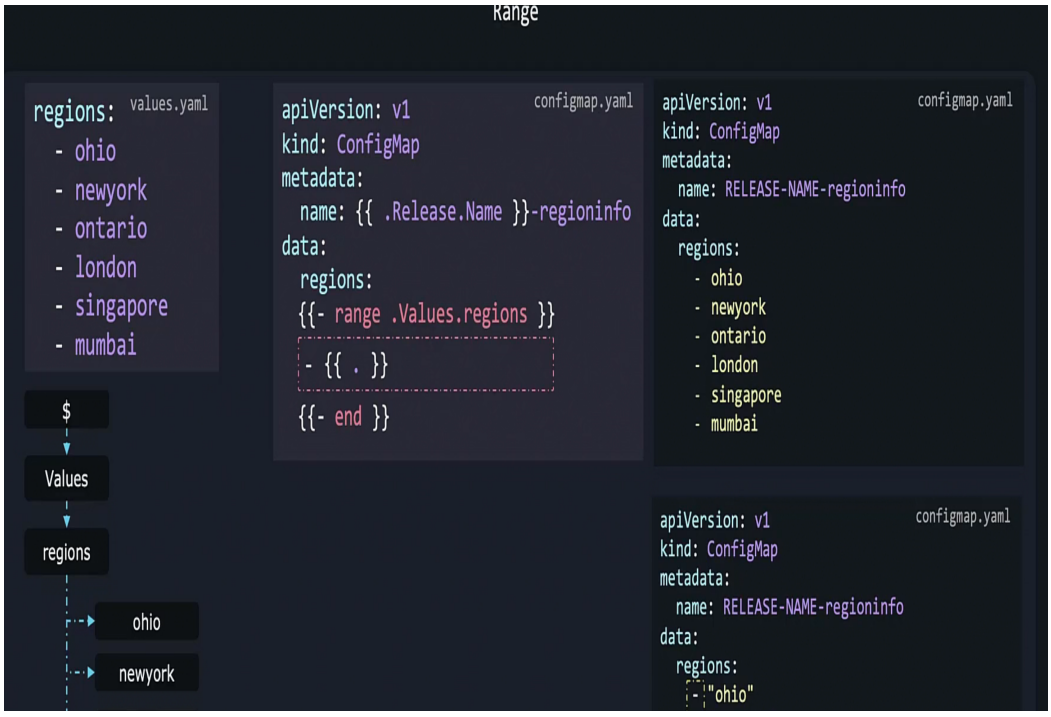


If you want to put a root scope inside, can use $

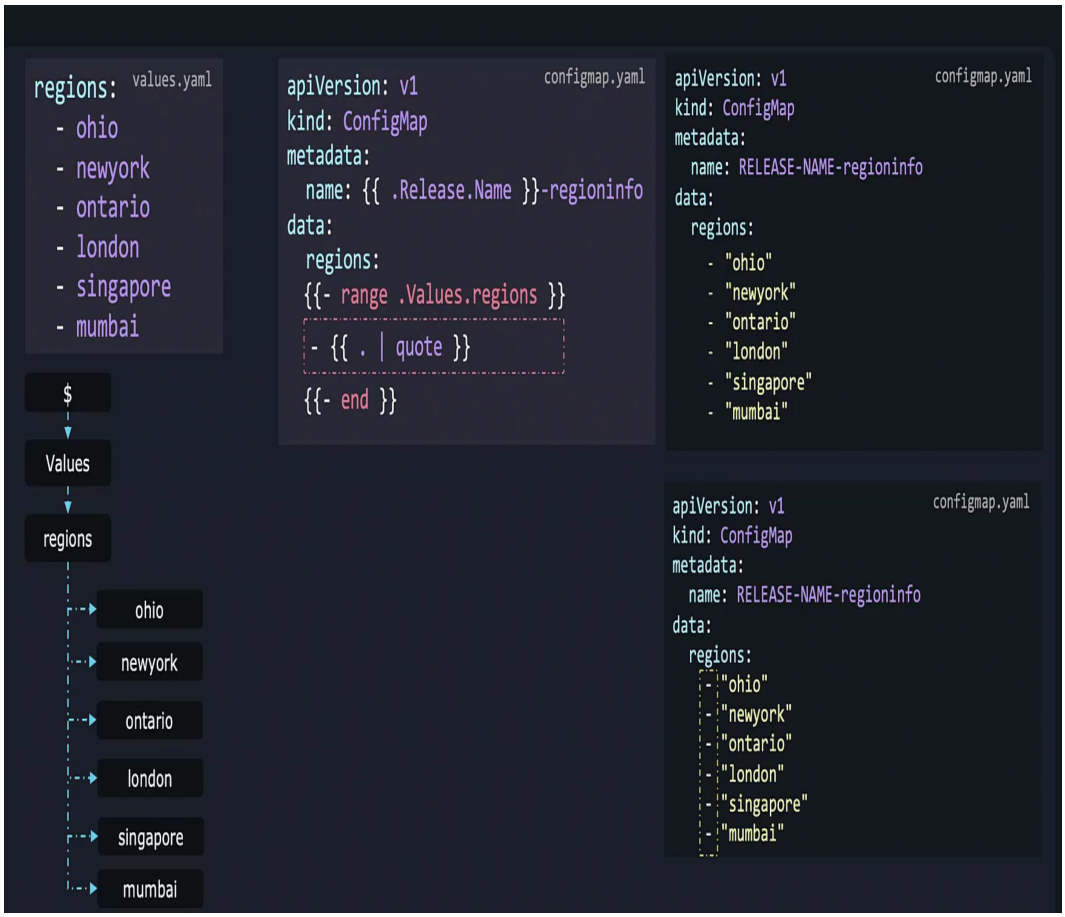
- if you want to access a variable that is not within the "with" block but outside of it
- use the $ to reference the root block



Ranges	A for loop, an example for config map



If you want to add quotes
- add {{ . | quote no}}



Named Templates

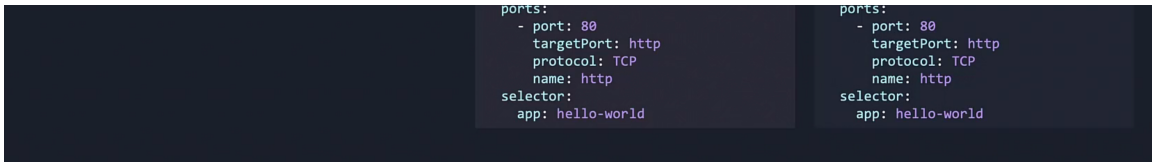
When there a few lines that are repeated many times in a file

We can use a helper file that starts with "_"
- e.g. "_named_template.tpl"

The _ tells helm to not take it as a standard template file - so helm won't make it a manifest file

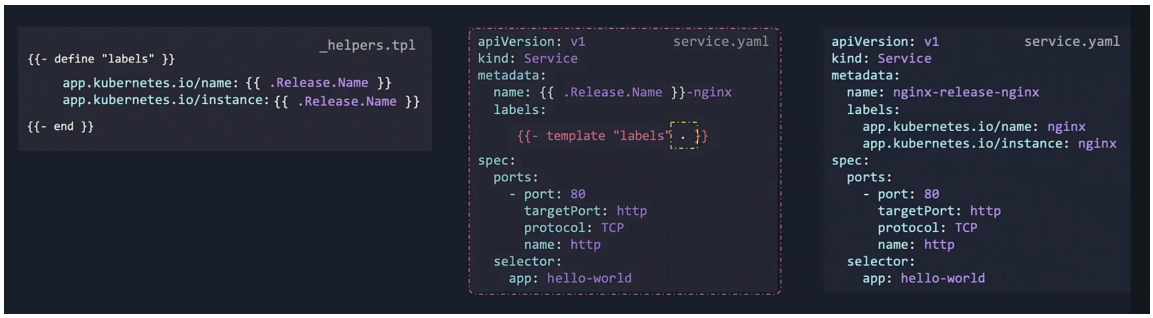
All files with _ will be skipped

This can be done as such



Or make it even better, add a variable in _helpers.tpl

- but need to add the "." in {{- template "labels" . }}
  - so we can add the current scope of values into the helper file
- need to ensure the spacing exist in the _helpers file, as it copies and pastes



However the spacing will give an issue as we might need extra indentation in another part of the file

- need to add " I indent 4 "
- but this don't work as template functions don't have the indentation function
- must use "include"

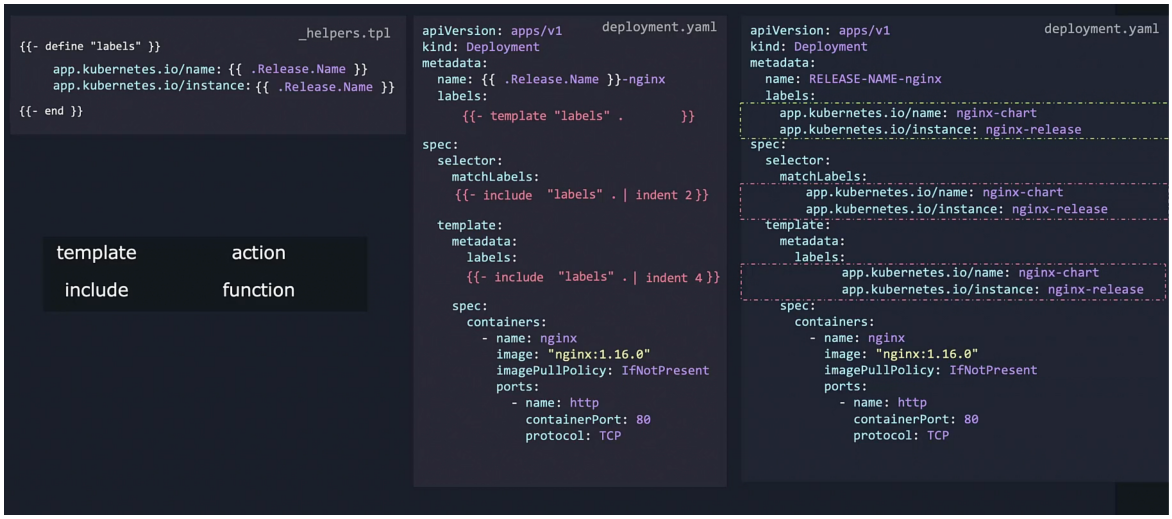


Chart Hooks

Useful in the following scenario

- whenever we do a "helm upgrade word-press bitnami/wordpress"
- we can trigger a database backup hook to backup the db before upgrade

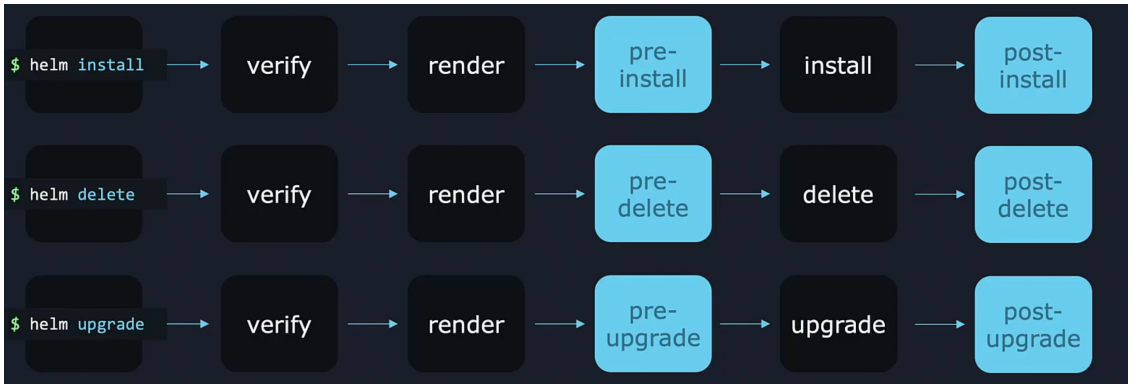
Normally when a user upgrades/install a helm file

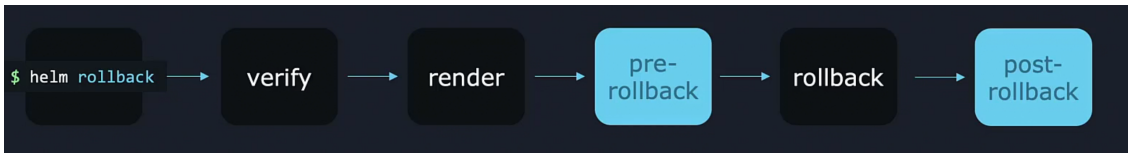
- he will verify and render a file then install the file on the cluster

In our case, we need to define the pre-upgrade/post-upgrade hook before/after the upgrade

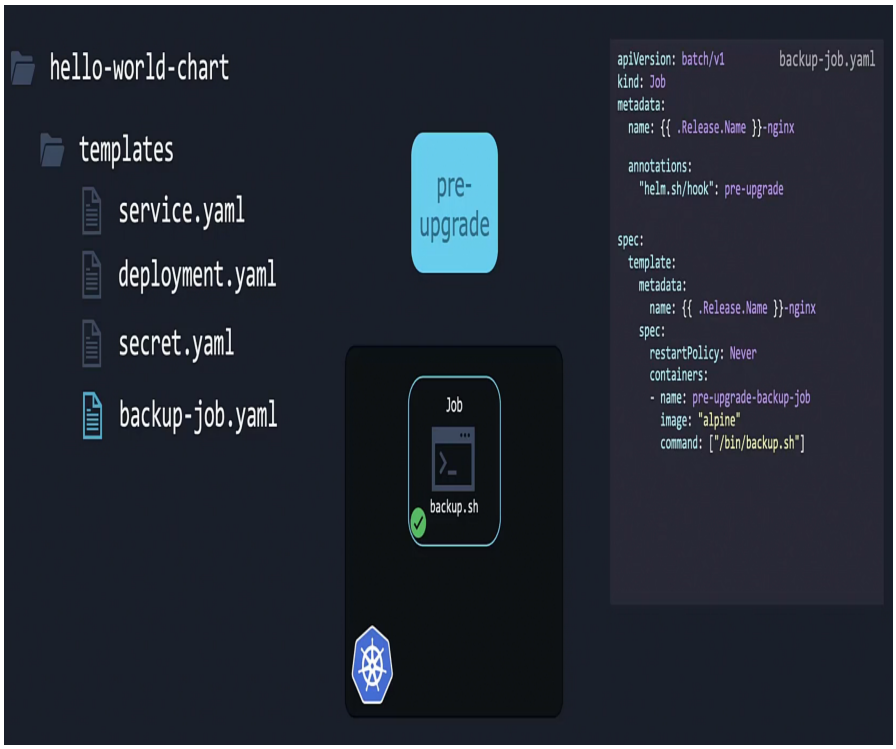
Many kinds of hook

- pre-install/post-install
- pre-delete/post-delete
- pre-upgrade/post-upgrade
- pre-rollback/post-rollback



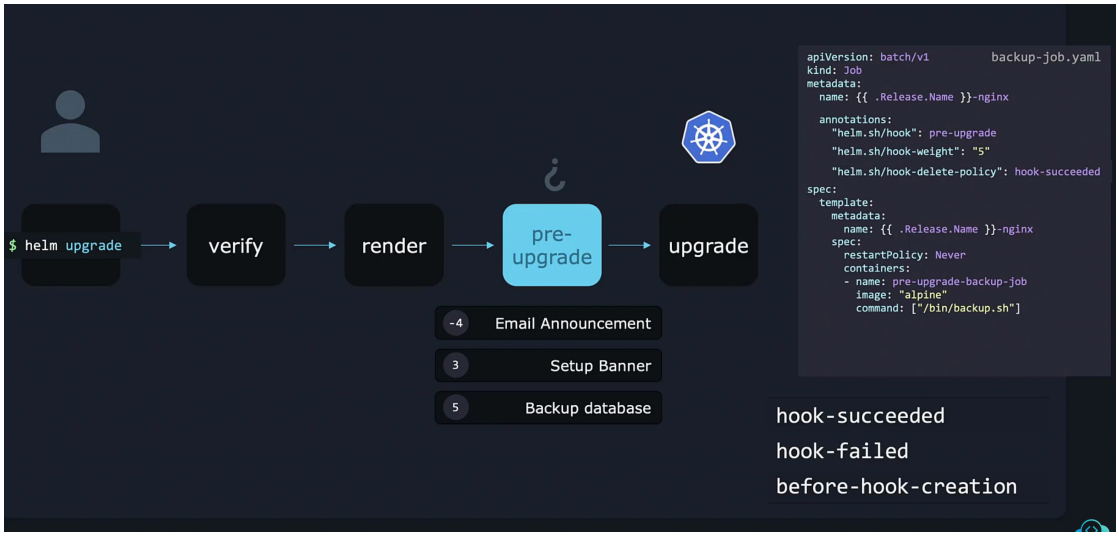


This hooks are ran as jobs - normally has a script that we run
- this jobs, we need to add the pre-upgrade annotation



These pre-upgrade stuff that can be ran
- setup banner
- backup database
- email annoucement

Can set weights on each hooks and the order that they should be run
- it will sort the jobs in ascending order and execute the jobs on that order
- need add to add the hook-weight annotation



Can add the annotation to delete the Job if the hook succeeds
- can let the Job still exist in the cluster if the hook fails so can debug
- before-hook-creation
  - whenever you run another hook, the pre-update-hook will delete the old one and create new one

Packaging and Signing Charts

To package a helm chart
- helm package ./nginx-chart

```
$ ls nginx-chart
  charts  Chart.yaml  templates  values.yaml README.md
  LICENSE


$ helm package ./nginx-chart
Successfully packaged chart and saved it to:
/vagrant/nginx-chart-0.1.0.tgz
```

The version is taken from the Chart.yaml file

The tgz extension refers to tar archive and can be opened via 7zip / any other utilities

Not recommended to just upload it, need to sign it for authenticity (due to hackers can place malicious files there)
- need to cryptographically sign charts and packages

Generate key
- gpg --quick-generate-key "John Smith"
- This command will generate a unique identifier for the key which could be uploaded to a public gpg key server, users can download it using this identifier to verify the signatures
- In production environment, can use
  - gpg --full-generate-key "John Smith"
  - or GnuGPGv2

Sign charts
- helm prefer the old format
- gpg --export-secret-keys >~/.gnupug/secret.gpg
- helm package --sign --key 'John Smith' --keyring ~/.gnupg/secring.gpg ./nginx-chart
- After signing, an additional file is created called the provisioning file
  - this will be stored in the .tgz.prov extension file
- dont understand

Verify signature
- helm verify ./nginx-chart-0.1.0.tgz

Real life
- users need download our public key
  - gpg --recv-keys --keyserver keyserver.ubuntu.com <key_id>
- verify the file
  - helm install --verify nginx-chart-0.1.0

Uploading Charts

After packaging and signing the chart, time to upload the chart online, so users can install it

- need to have the 2 files
  - zipped file
  - provn file (gotten after signing)

```
$ ls
  nginx-chart  nginx-chart-0.1.0.tgz  nginx-chart-0.1.0.tgz.prov

$ mkdir nginx-chart-files


$ cp nginx-chart-0.1.0.tgz  nginx-chart-0.1.0.tgz.provn ginx-chart-files/


$ helm repo index nginx-chart-files/ --url https://example.com/charts


$ ls nginx-chart-files
  index.yaml  nginx-chart-0.1.0.tgz  nginx-chart-0.1.0.tgz.prov
```

For upload, can just upload to google storage / aws buckets / etc
- can just share the URL to whoever wants to download the chart
- they just need to
- helm repo add xxx-chart <url>
- helm repo list - to confirm repo is added
- helm install <release> <chart_name>