Helm KodeKloud

Sunday, 11 June 2023 10:06 PM

Section | **Description**

What is Helm?

Helm - package manager for kubernetes

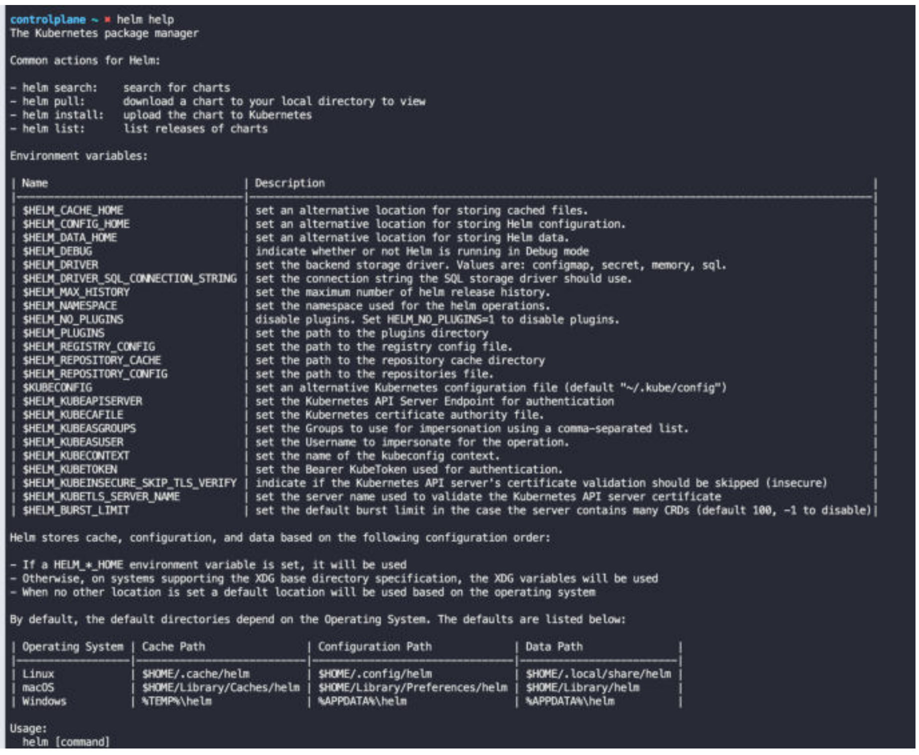
Why helm might be beneficial?
- For an application, we might need multiple YAML files e.g. deployment, secrets, etc
- We need custom parameters for the values in the fields of each yaml file
- We might also need versioning
- We can upgrade our application with a single command
  - helm upgrade wordpress
- 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

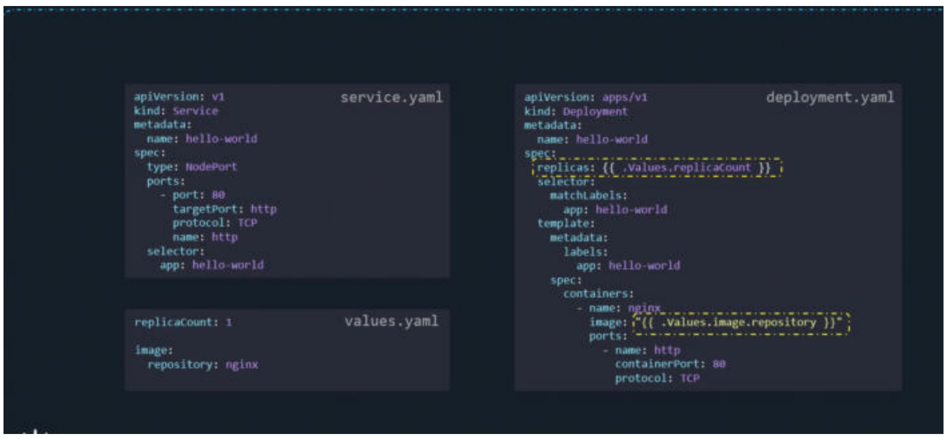
Can configure your helm parameters via the environment variables



```
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



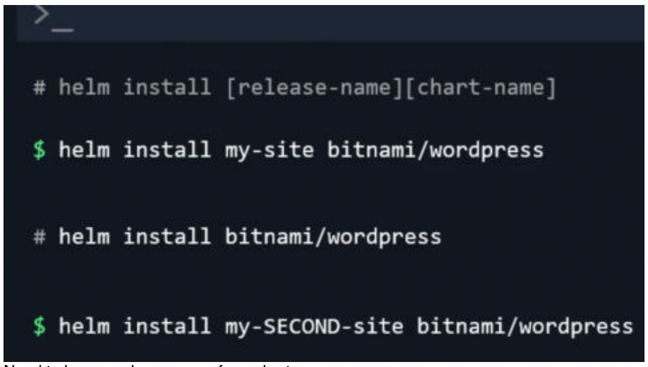
service.yaml
```yaml
apiVersion: v1
kind: Service
metadata:
  name: hello-world
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: http
      protocol: TCP
      name: http
  selector:
    app: hello-world
```

values.yaml
```yaml
replicaCount: 1

image:
  repository: nginx
```

deployment.yaml
```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
```

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

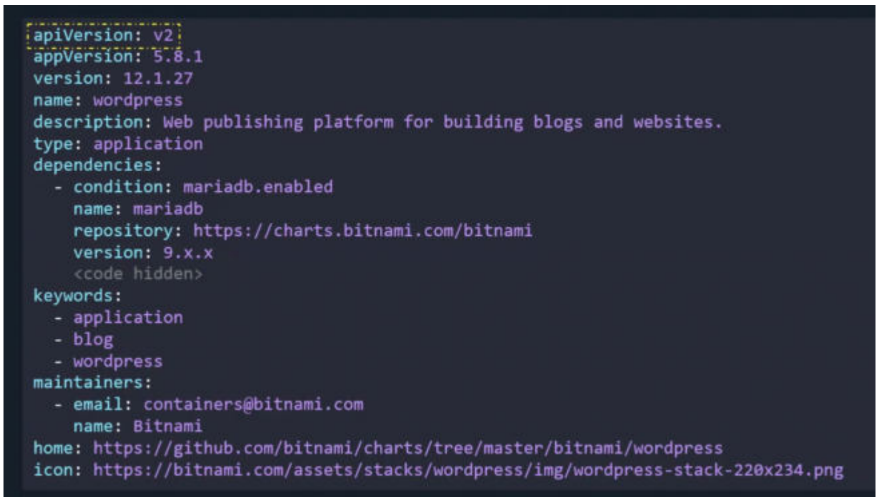


```
>_

# 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



```
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
```

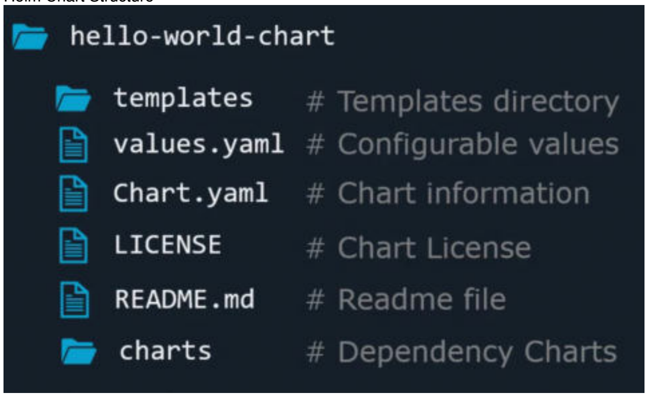
apiVersion: v1 --> if you use helm2, but if helm3, use v2
appVersion: --> version of the application thats inside of this chart - version of wordpress that is being deployed (for informational purposes only)
version: version of the chart itself
type: 2 types of charts - application/libraray
- application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

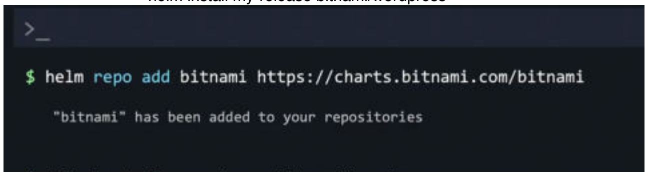


```
hello-world-chart
    templates      # Templates directory
    values.yaml    # Configurable values
    Chart.yaml     # Chart information
    LICENSE        # Chart License
    README.md      # Readme file
    charts         # Dependency Charts
```

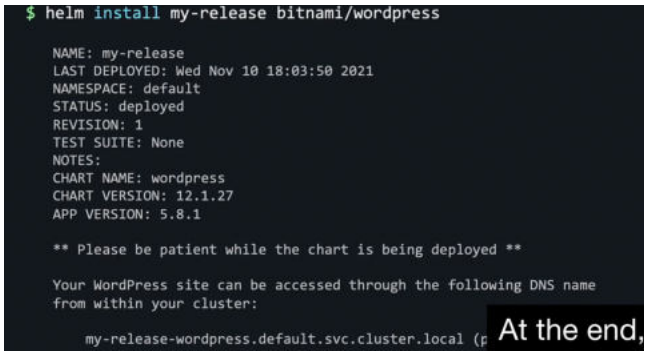
Working with Helm: basics

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
```



```
$ helm install my-release bitnami/wordpress

    NAME: my-release
    LAST DEPLOYED: Wed Nov 10 18:03:50 2021
    NAMESPACE: default
    STATUS: deployed
    REVISION: 1
    TEST SUITE: None
    NOTES:
    CHART NAME: wordpress
    CHART VERSION: 12.1.27
    APP VERSION: 5.8.1

    ** Please be patient while the chart is being deployed **

    Your WordPress site can be accessed through the following DNS name
    from within your cluster:

        my-release-wordpress.default.svc.cluster.local (p
```

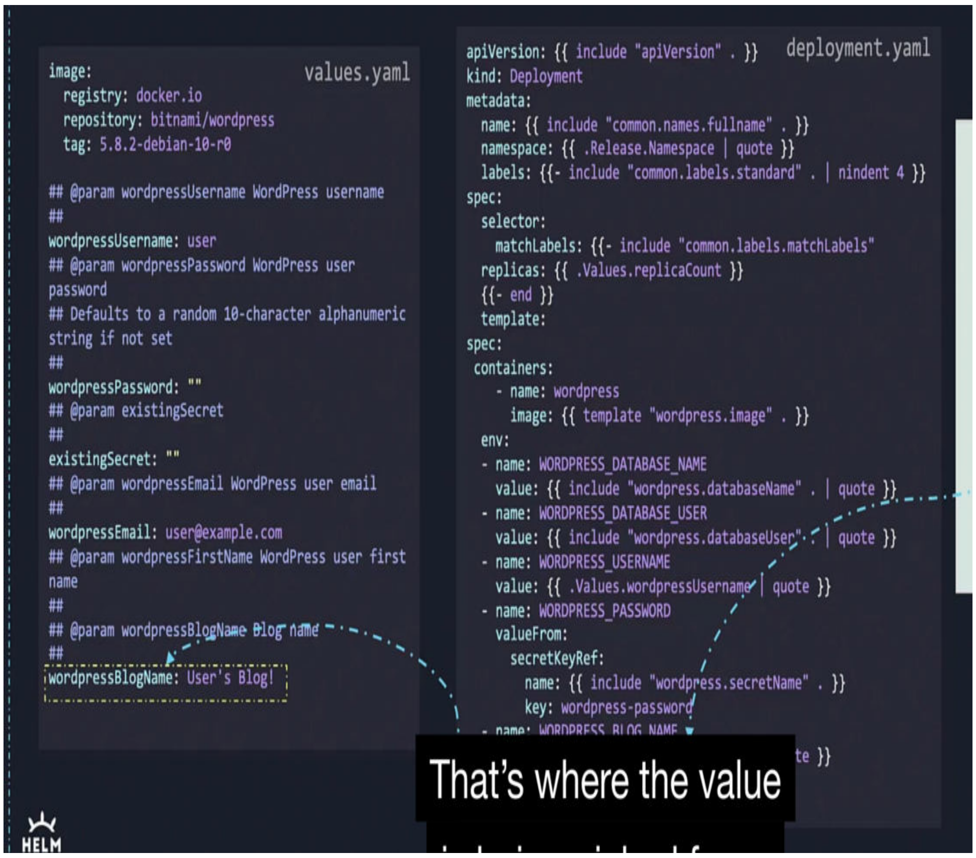
Once a chart is deployed, it is deployed as a release
- `helm list`
- when we want to delete, we don't have to remove them from the cluster one by one
    - we can just *helm uninstall my-release*

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.

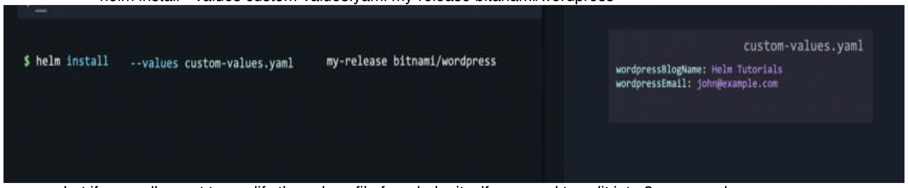
We may not want the word press site to have the default name
- the wordpress deployment file contains templating values for the environment variables which will affect the wordpress name



```
values.yaml
image:
  registry: docker.io
  repository: bitnami/wordpress
  tag: 5.8.2-debian-10-r0

## @param wordpressUsername WordPress username
##
wordpressUsername: user
## @param wordpressPassword WordPress user
password
## Defaults to a random 10-character alphanumeric
string if not set
##
wordpressPassword: ""
## @param existingSecret
##
existingSecret: ""
## @param wordpressEmail WordPress user email
##
wordpressEmail: user@example.com
## @param wordpressFirstName WordPress user first
name
##
## @param wordpressBlogName Blog name
##
wordpressBlogName: User's Blog!
```

```
apiVersion: {{ include "apiVersion" . }}    deployment.yaml
kind: Deployment
metadata:
  name: {{ include "common.names.fullname" . }}
  namespace: {{ .Release.Namespace | quote }}
  labels: {{- include "common.labels.standard" . | nindent 4 }}
spec:
  selector:
    matchLabels: {{- include "common.labels.matchLabels"
  replicas: {{ .Values.replicaCount }}
  {{- end }}
  template:
spec:
 containers:
   - name: wordpress
     image: {{ template "wordpress.image" . }}
  env:
  - name: WORDPRESS_DATABASE_NAME
    value: {{ include "wordpress.databaseName" . | quote }}
  - name: WORDPRESS_DATABASE_USER
    value: {{ include "wordpress.databaseUser" . | quote }}
  - name: WORDPRESS_USERNAME
    value: {{ .Values.wordpressUsername | quote }}
  - name: WORDPRESS_PASSWORD
    valueFrom:
      secretKeyRef:
        name: {{ include "wordpress.secretName" . }}
        key: wordpress-password
  - name: WORDPRESS_BLOG_NAME
                                   te }}
```

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



```
$ helm install   --values custom-values.yaml   my-release bitnami/wordpress
```

```
custom-values.yaml
wordpressBlogName: Helm Tutorials
wordpressEmail: john@example.com
```

- 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

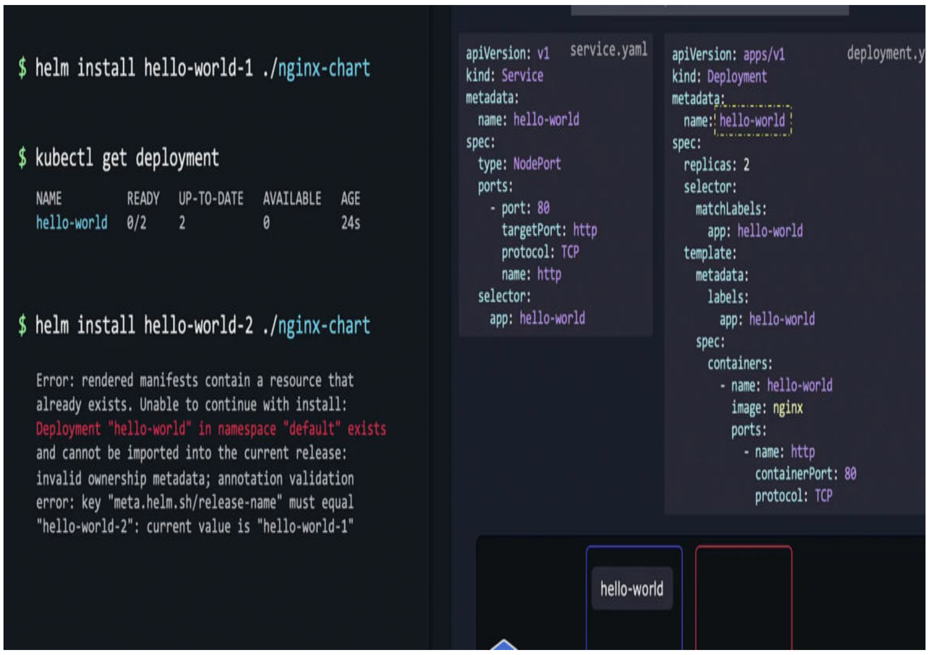
helm list
helm history nginx-release --> can see a lot of useful releases
helm rollback nginx-release <revision number>

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/



```
$ helm install hello-world-1 ./nginx-chart

$ kubectl get deployment
  NAME         READY   UP-TO-DATE   AVAILABLE   AGE
  hello-world  0/2     2            0           24s

$ helm install hello-world-2 ./nginx-chart

  Error: rendered manifests contain a resource that
  already exists. Unable to continue with install:
  Deployment "hello-world" in namespace "default" exists
  and cannot be imported into the current release:
  invalid ownership metadata; annotation validation
  error: key "meta.helm.sh/release-name" must equal
  "hello-world-2": current value is "hello-world-1"
```

```yaml
# 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
```

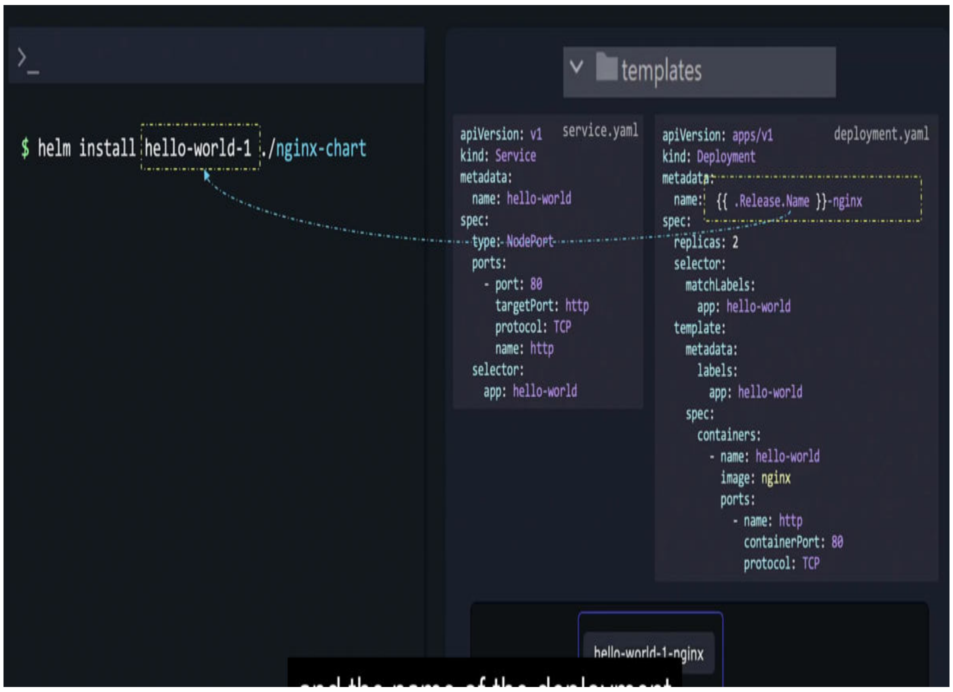
```yaml
# deployment.y
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-world
spec:
  replicas: 2
  selector:
    matchLabels:
      app: hello-world
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
        - name: hello-world
          image: nginx
          ports:
            - name: http
              containerPort: 80
              protocol: TCP
```

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



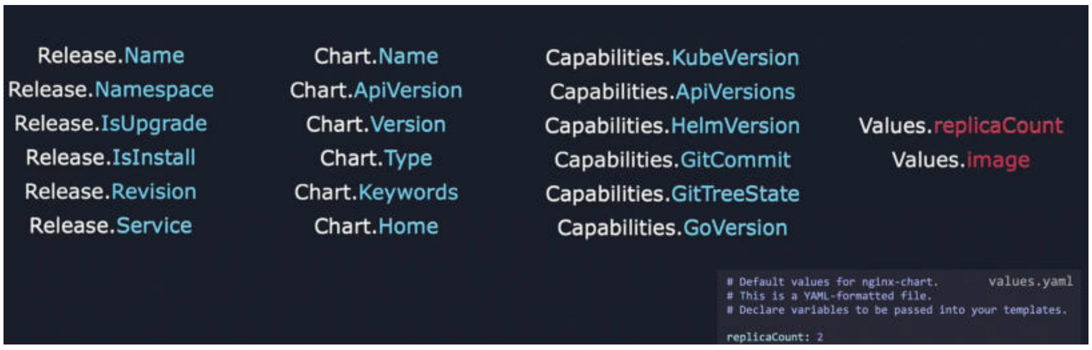
```
>_

$ helm install hello-world-1 ./nginx-chart
```

```yaml
# 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
```

```yaml
# deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name }}-nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: hello-world
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
        - name: hello-world
          image: nginx
          ports:
            - name: http
              containerPort: 80
              protocol: TCP
```

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

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

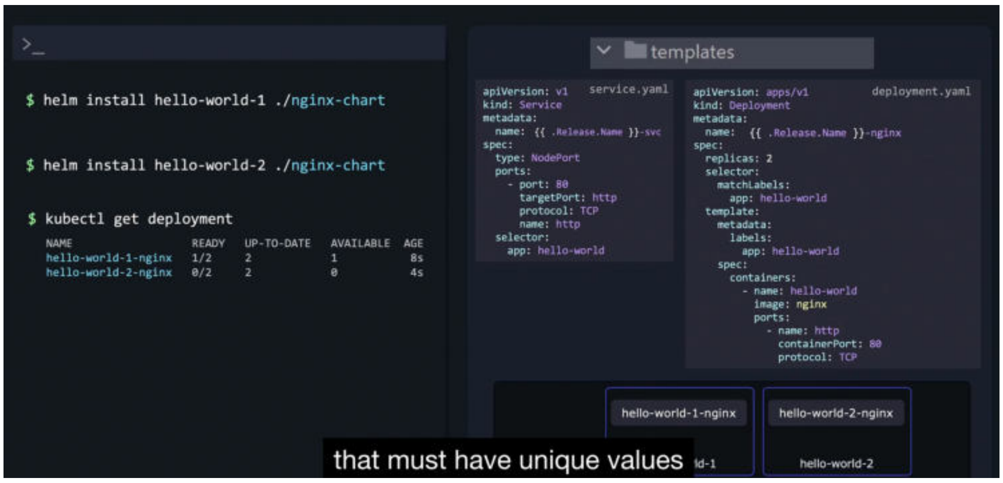


Release.Name	Chart.Name	Capabilities.KubeVersion	
Release.Namespace	Chart.ApiVersion	Capabilities.ApiVersions	
Release.IsUpgrade	Chart.Version	Capabilities.HelmVersion	Values.replicaCount
Release.IsInstall	Chart.Type	Capabilities.GitCommit	Values.image
Release.Revision	Chart.Keywords	Capabilities.GitTreeState	
Release.Service	Chart.Home	Capabilities.GoVersion	

```yaml
# values.yaml
# Default values for nginx-chart.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

replicaCount: 2
```

```
image: nginx
```

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



```
$ helm install hello-world-1 ./nginx-chart

$ helm install hello-world-2 ./nginx-chart

$ kubectl get deployment
  NAME                READY   UP-TO-DATE   AVAILABLE   AGE
  hello-world-1-nginx 1/2     2            1           8s
  hello-world-2-nginx 0/2     2            0           4s
```

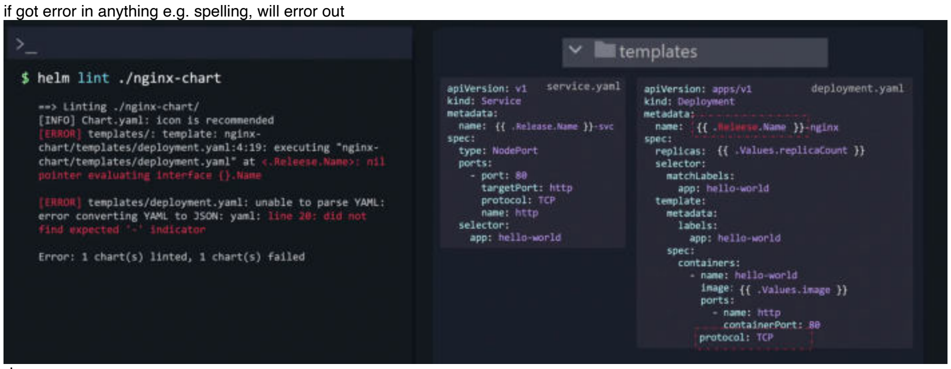
templates

service.yaml
```yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}-svc
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: http
      protocol: TCP
      name: http
  selector:
    app: hello-world
```

deployment.yaml
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name }}-nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: hello-world
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
        - name: hello-world
          image: nginx
          ports:
            - name: http
              containerPort: 80
              protocol: TCP
```

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
  - 

```
$ helm lint ./nginx-chart

  ==> Linting ./nginx-chart/
  [INFO] Chart.yaml: icon is recommended
  [ERROR] templates/: template: nginx-
  chart/templates/deployment.yaml:4:19: executing "nginx-
  chart/templates/deployment.yaml" at <.Releese.Name>: nil
  pointer evaluating interface {}.Name

  [ERROR] templates/deployment.yaml: unable to parse YAML:
  error converting YAML to JSON: yaml: line 20: did not
  find expected '-' indicator

  Error: 1 chart(s) linted, 1 chart(s) failed
```

templates

service.yaml
```yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}-svc
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: http
      protocol: TCP
      name: http
  selector:
    app: hello-world
```

deployment.yaml
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Releese.Name }}-nginx
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      app: hello-world
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
        - name: hello-world
          image: {{ .Values.image }}
          ports:
            - name: http
              containerPort: 80
           protocol: TCP
```

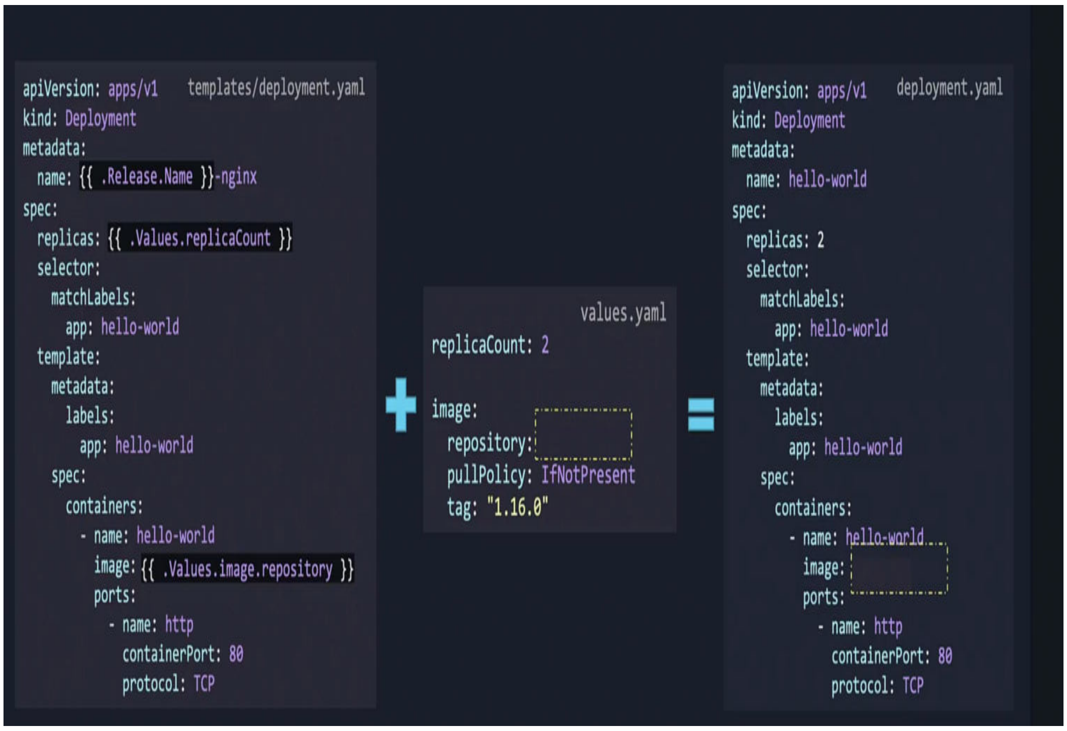
- Template
  - helm template <chart_name>
  - renders the output as per the values.yaml into the templates/
  - if got error, will error out
- Dry Run
  - the two above can help to catch many errors but cannot catch
    - typo of containers vs container
    - only kubernetes can tell us if this is wrong
  - 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?

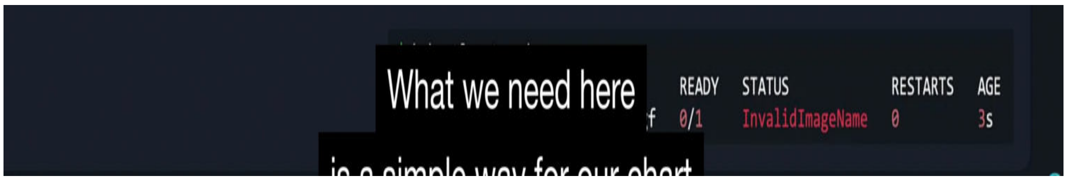
That will create the output file without that field



templates/deployment.yaml
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name }}-nginx
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      app: hello-world
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
        - name: hello-world
          image: {{ .Values.image.repository }}
          ports:
            - name: http
              containerPort: 80
              protocol: TCP
```

values.yaml
```yaml
replicaCount: 2

image:
  repository:
  pullPolicy: IfNotPresent
  tag: "1.16.0"
```

deployment.yaml
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-world
spec:
  replicas: 2
  selector:
    matchLabels:
      app: hello-world
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
        - name: hello-world
          image:
          ports:
            - name: http
              containerPort: 80
              protocol: TCP
```

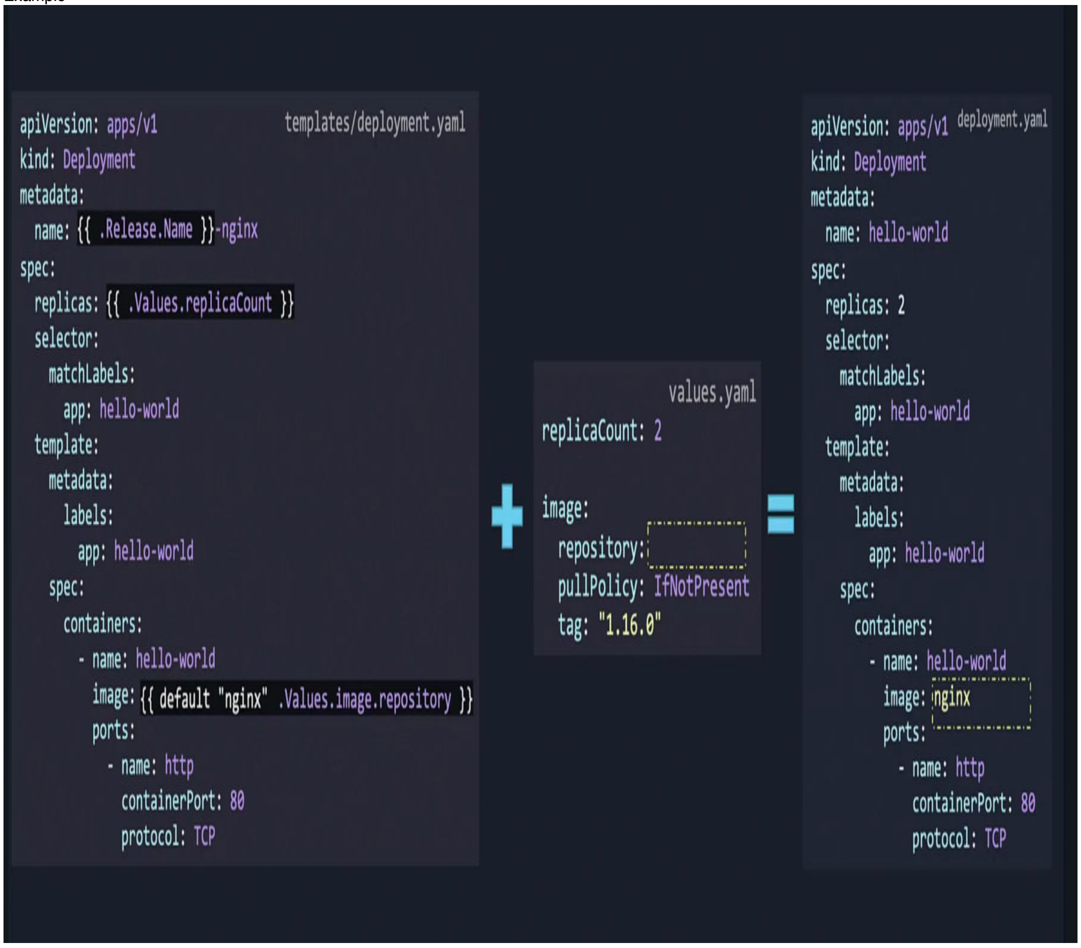


```
                         READY   STATUS             RESTARTS   AGE
                       f 0/1     InvalidImageName   0          3s
```

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



templates/deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name }}-nginx
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      app: hello-world
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
        - name: hello-world
          image: {{ default "nginx" .Values.image.repository }}
          ports:
            - name: http
              containerPort: 80
              protocol: TCP
```

+

values.yaml

```
replicaCount: 2

image:
  repository:
  pullPolicy: IfNotPresent
  tag: "1.16.0"
```

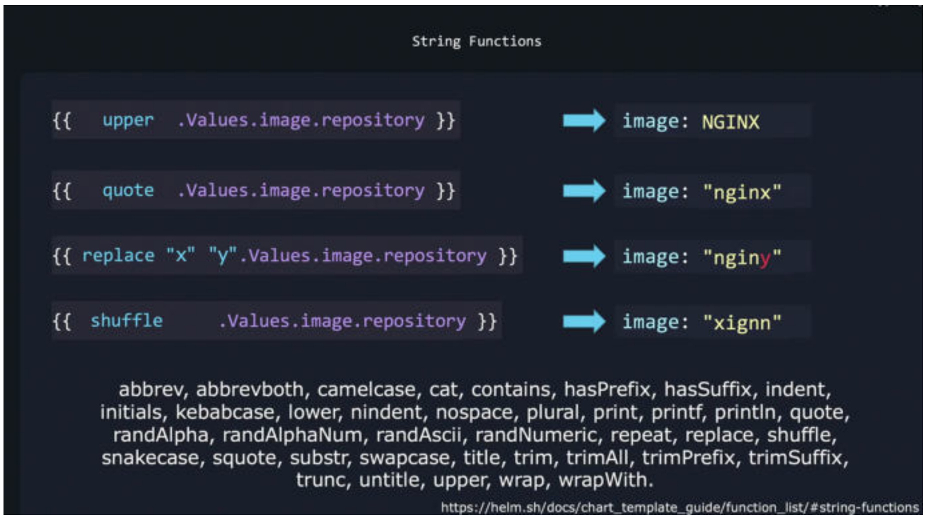
=

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-world
spec:
  replicas: 2
  selector:
    matchLabels:
      app: hello-world
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
        - name: hello-world
          image: nginx
          ports:
            - name: http
              containerPort: 80
              protocol: TCP
```

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/

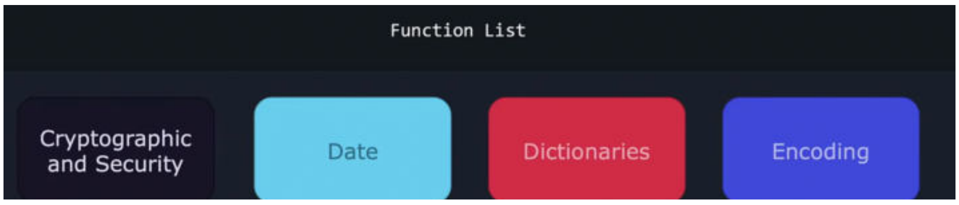


String Functions

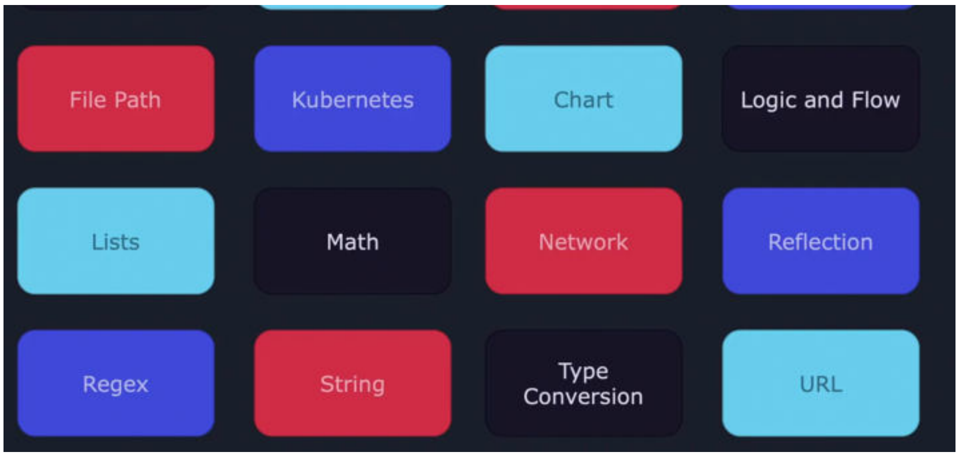
```
{{ upper .Values.image.repository }}              → image: NGINX
{{ quote .Values.image.repository }}              → image: "nginx"
{{ replace "x" "y" .Values.image.repository }}    → image: "nginy"
{{ shuffle .Values.image.repository }}            → image: "xignn"
```

abbrev, abbrevboth, camelcase, cat, contains, hasPrefix, hasSuffix, indent, initials, kebabcase, lower, nindent, nospace, plural, print, printf, println, quote, randAlpha, randAlphaNum, randAscii, randNumeric, repeat, replace, shuffle, snakecase, squote, substr, swapcase, title, trim, trimAll, trimPrefix, trimSuffix, trunc, untitle, upper, wrap, wrapWith.

https://helm.sh/docs/chart_template_guide/function_list/#string-functions



Function List

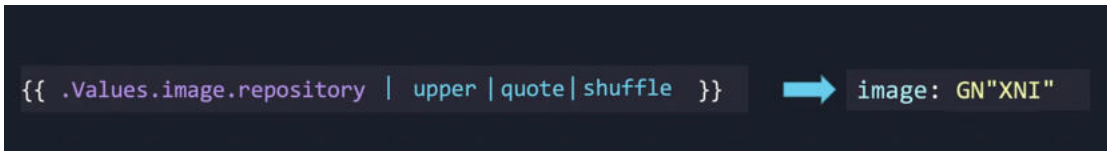


Pipelines

You can pipe functions like this

- the functions are upper/quote/shuffle

Chaining these functions with a pipe is known as a pipeline

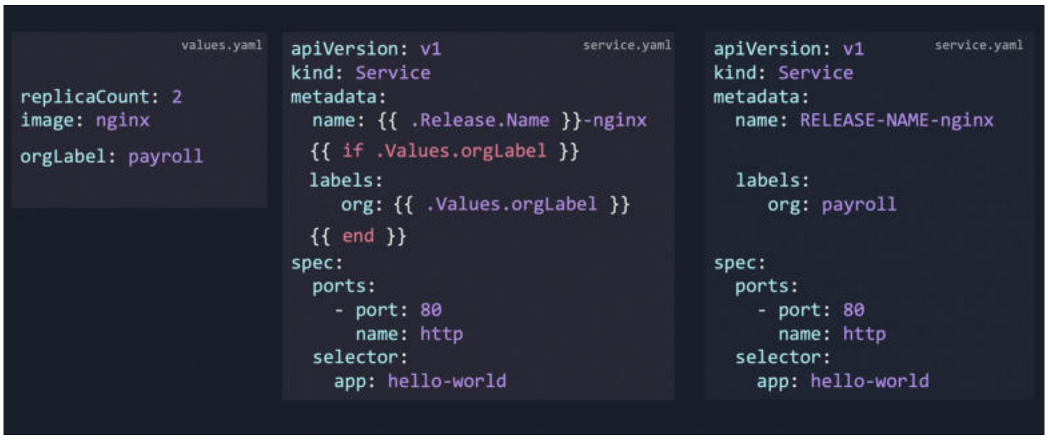


```
{{ .Values.image.repository | upper | quote | shuffle }}
```

```
image: GN"XNI"
```

Conditionals

If statements in the helm file



values.yaml

```
replicaCount: 2
image: nginx

orgLabel: payroll
```

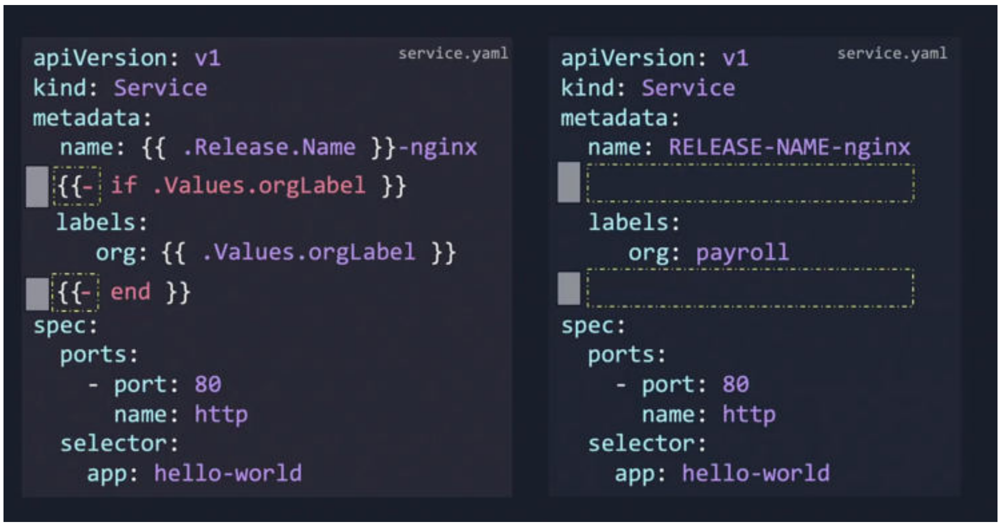
service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}-nginx
  {{ if .Values.orgLabel }}
  labels:
     org: {{ .Values.orgLabel }}
  {{ end }}
spec:
  ports:
    - port: 80
      name: http
  selector:
    app: hello-world
```

service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: RELEASE-NAME-nginx

  labels:
     org: payroll

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

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

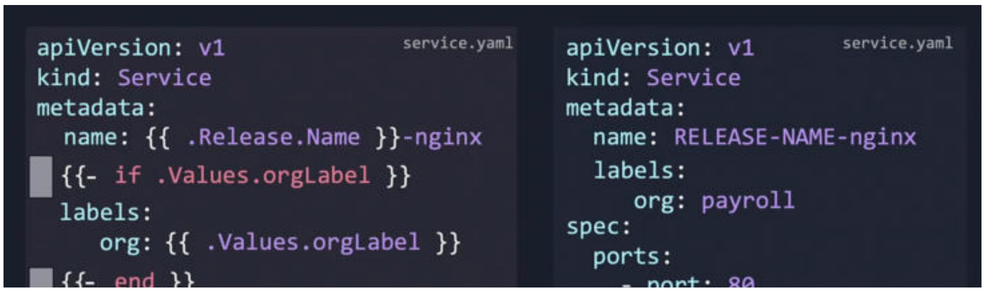


service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}-nginx
 {{- if .Values.orgLabel }}
  labels:
     org: {{ .Values.orgLabel }}
 {{- end }}
spec:
  ports:
    - port: 80
      name: http
  selector:
    app: hello-world
```

service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: RELEASE-NAME-nginx

  labels:
     org: payroll

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

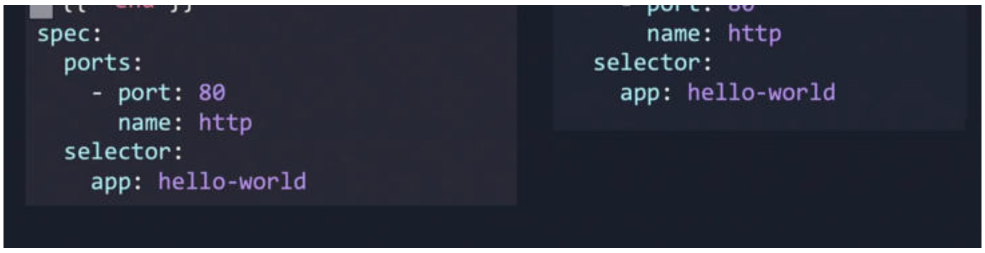


service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}-nginx
 {{- if .Values.orgLabel }}
  labels:
     org: {{ .Values.orgLabel }}
 {{- end }}
```

service.yaml

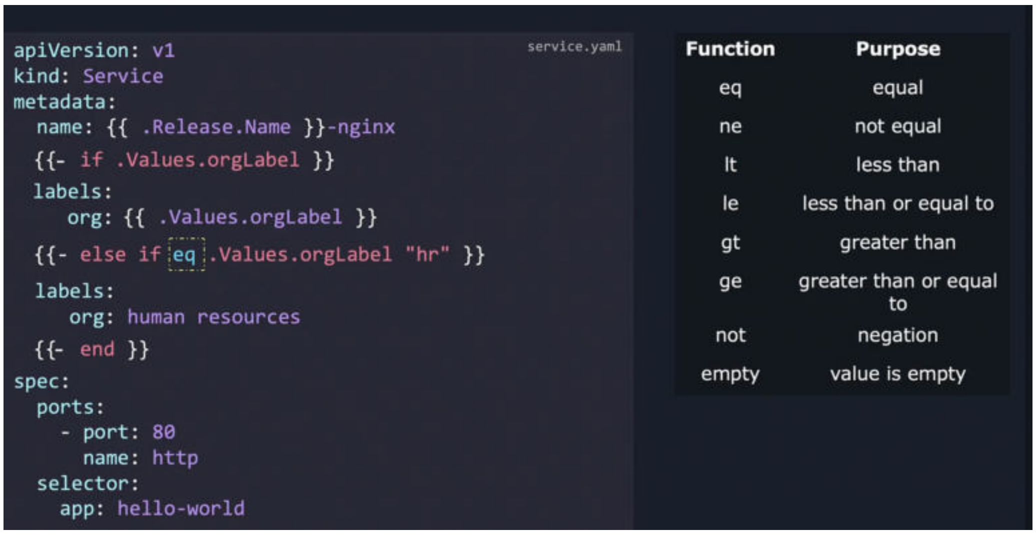
```
apiVersion: v1
kind: Service
metadata:
  name: RELEASE-NAME-nginx
  labels:
     org: payroll
spec:
  ports:
```



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

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

else if statements

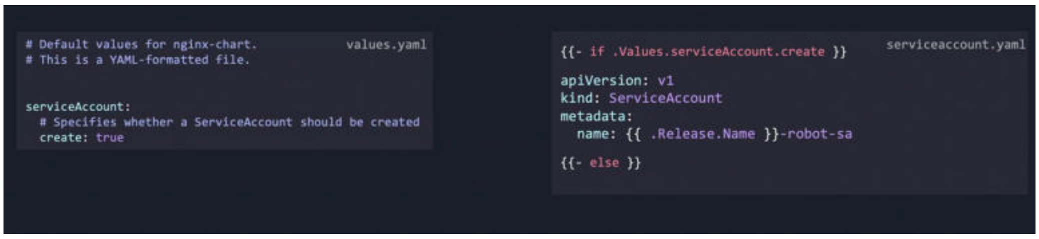


```
apiVersion: v1                                    service.yaml
kind: Service
metadata:
  name: {{ .Release.Name }}-nginx
  {{- if .Values.orgLabel }}
  labels:
     org: {{ .Values.orgLabel }}
  {{- else if eq .Values.orgLabel "hr" }}
  labels:
     org: human resources
  {{- end }}
spec:
  ports:
    - 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,

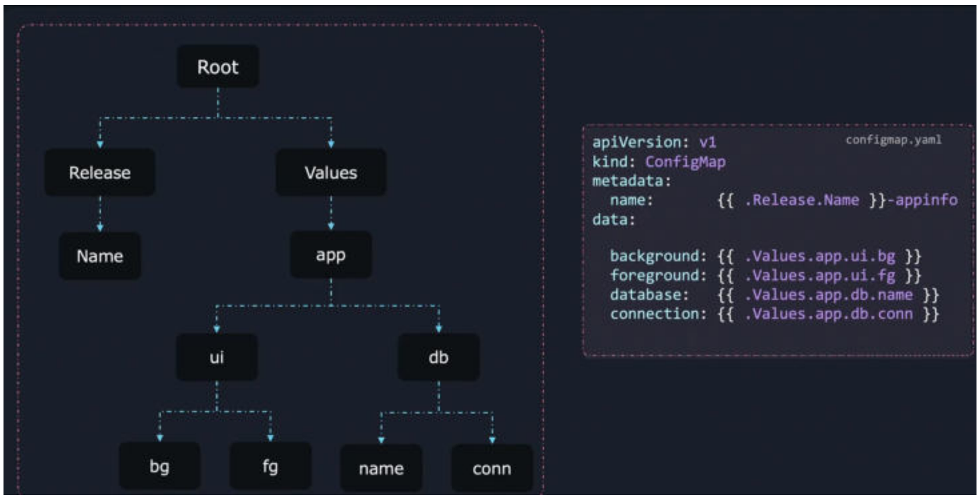
- for this can set up illumio helm enable daemonset, if "enable" then we put on all hosts



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


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

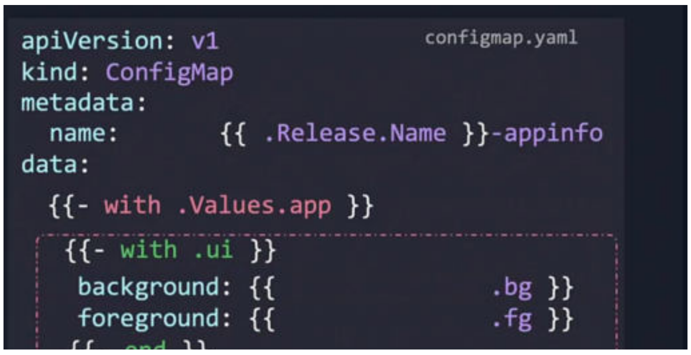
```
{{- if .Values.serviceAccount.create }}          serviceaccount.yaml

apiVersion: v1
kind: ServiceAccount
metadata:
  name: {{ .Release.Name }}-robot-sa

{{- else }}
```

With Blocks

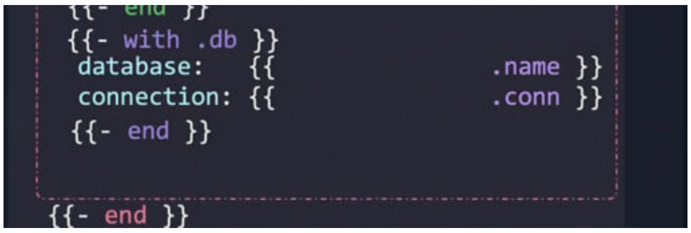


```
apiVersion: v1                                configmap.yaml
kind: ConfigMap
metadata:
  name:       {{ .Release.Name }}-appinfo
data:

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

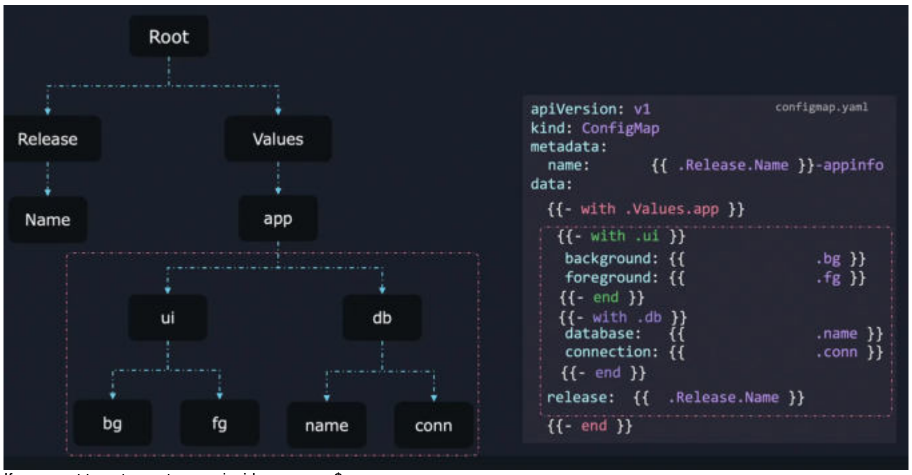
We see that there are duplications for multiple .Values.app., we can change to



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

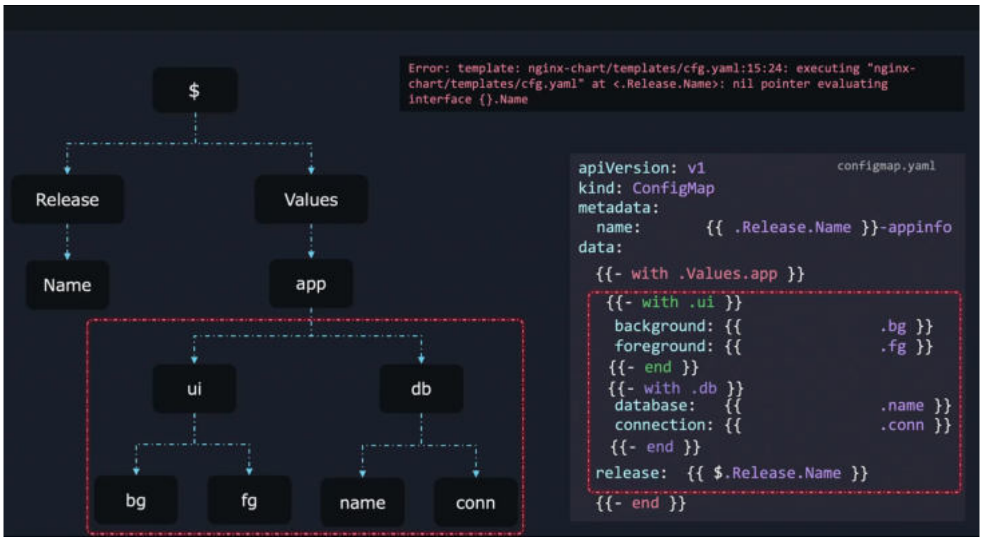


```
  {{- end }}
  {{- with .db }}
   database:    {{              .name }}
   connection: {{               .conn }}
  {{- end }}

{{- end }}
```



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

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

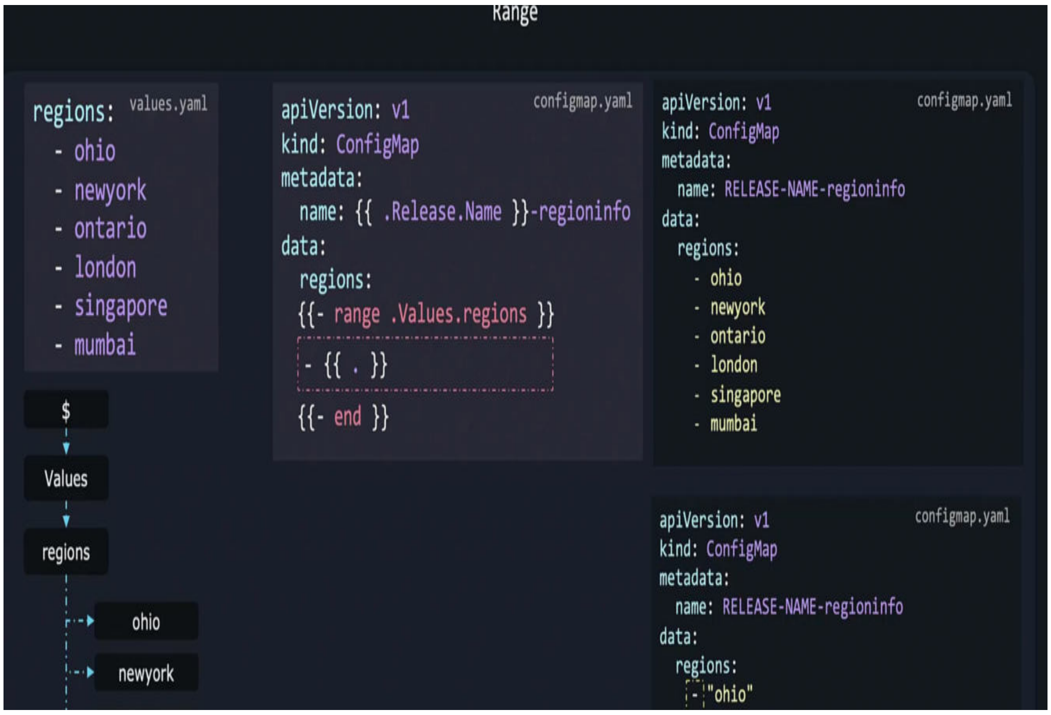


```
Error: template: nginx-chart/templates/cfg.yaml:15:24: executing "nginx-
chart/templates/cfg.yaml" at <.Release.Name>: nil pointer evaluating
interface {}.Name
```

```
apiVersion: v1                              configmap.yaml
kind: ConfigMap
metadata:
  name:        {{ .Release.Name }}-appinfo
data:
  {{- with .Values.app }}
   {{- with .ui }}
    background: {{              .bg }}
    foreground: {{              .fg }}
   {{- end }}
   {{- with .db }}
    database:    {{             .name }}
    connection: {{              .conn }}
   {{- end }}
  release:  {{ $.Release.Name }}
  {{- end }}
```

Ranges

A for loop, an example for config map

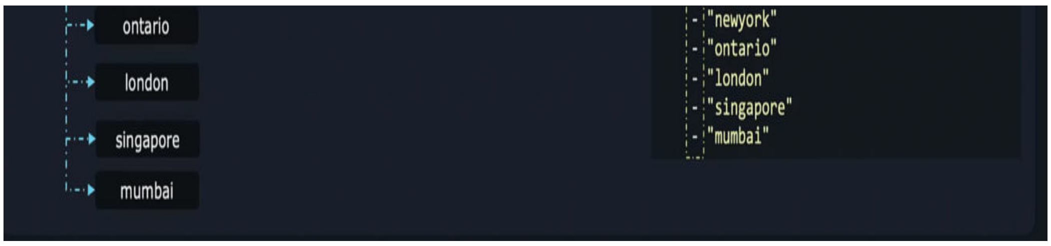


```
regions:                    values.yaml
  - ohio
  - newyork
  - ontario
  - london
  - singapore
  - mumbai
```

```
apiVersion: v1                configmap.yaml
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-regioninfo
data:
  regions:
  {{- range .Values.regions }}
  - {{ . }}
  {{- end }}
```

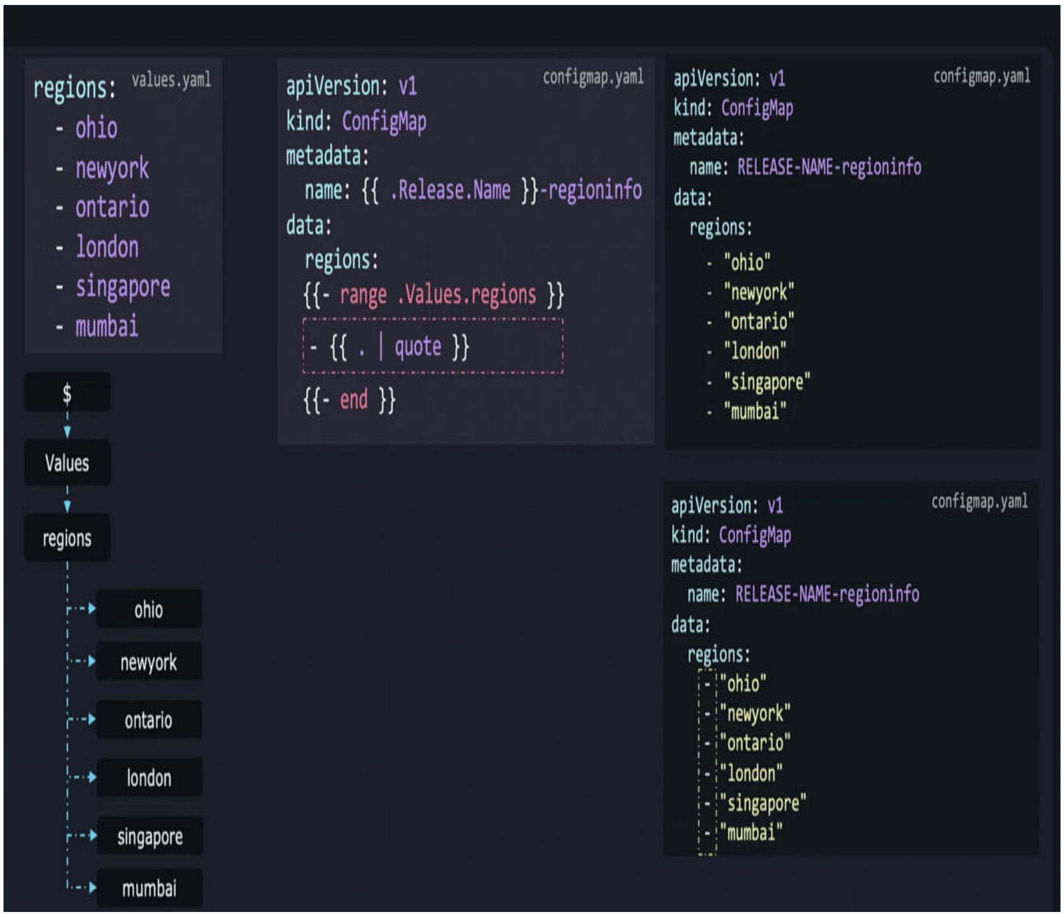
```
apiVersion: v1                configmap.yaml
kind: ConfigMap
metadata:
  name: RELEASE-NAME-regioninfo
data:
  regions:
    - ohio
    - newyork
    - ontario
    - london
    - singapore
    - mumbai
```

```
apiVersion: v1                configmap.yaml
kind: ConfigMap
metadata:
  name: RELEASE-NAME-regioninfo
data:
  regions:
  - "ohio"
```



```
- "newyork"
- "ontario"
- "london"
- "singapore"
- "mumbai"
```

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



values.yaml
```
regions:
  - ohio
  - newyork
  - ontario
  - london
  - singapore
  - mumbai
```

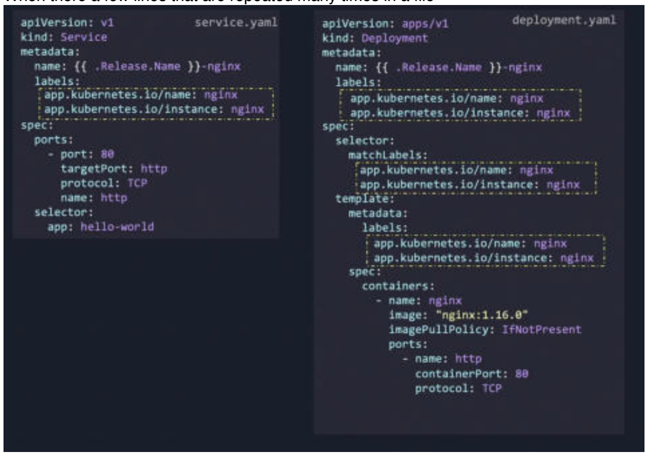
configmap.yaml
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-regioninfo
data:
  regions:
  {{- range .Values.regions }}
  - {{ . | quote }}
  {{- end }}
```

configmap.yaml
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: RELEASE-NAME-regioninfo
data:
  regions:
    - "ohio"
    - "newyork"
    - "ontario"
    - "london"
    - "singapore"
    - "mumbai"
```

configmap.yaml
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: RELEASE-NAME-regioninfo
data:
  regions:
  - "ohio"
  - "newyork"
  - "ontario"
  - "london"
  - "singapore"
  - "mumbai"
```

Named Templates

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



service.yaml
```
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}-nginx
  labels:
    app.kubernetes.io/name: nginx
    app.kubernetes.io/instance: nginx
spec:
  ports:
    - port: 80
      targetPort: http
      protocol: TCP
      name: http
  selector:
    app: hello-world
```

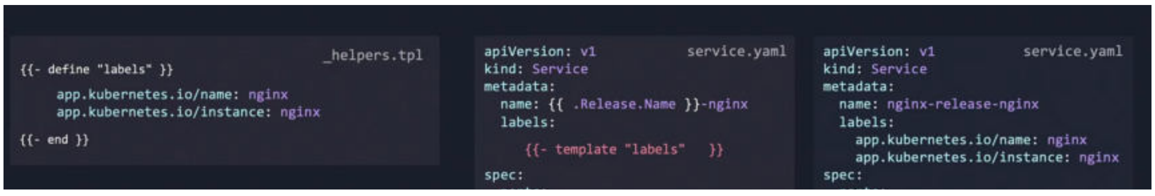
deployment.yaml
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name }}-nginx
  labels:
    app.kubernetes.io/name: nginx
    app.kubernetes.io/instance: nginx
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: nginx
      app.kubernetes.io/instance: nginx
  template:
    metadata:
      labels:
        app.kubernetes.io/name: nginx
        app.kubernetes.io/instance: nginx
    spec:
      containers:
        - name: nginx
          image: "nginx:1.16.0"
          imagePullPolicy: IfNotPresent
          ports:
            - name: http
              containerPort: 80
              protocol: TCP
```

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

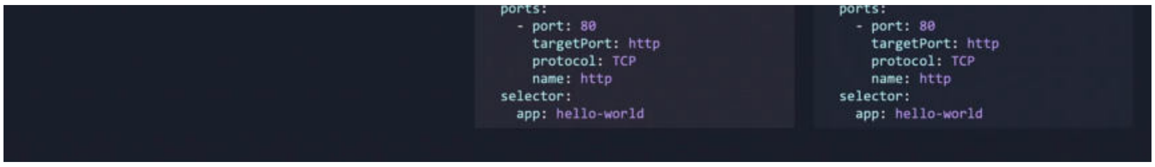
This can be done as such



_helpers.tpl
```
{{- define "labels" }}
    app.kubernetes.io/name: nginx
    app.kubernetes.io/instance: nginx
{{- end }}
```

service.yaml
```
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}-nginx
  labels:
    {{- template "labels"  }}
spec:
```

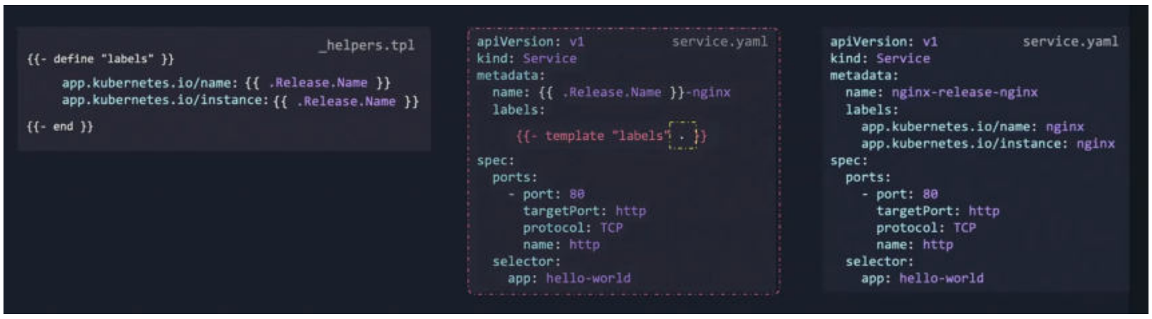
service.yaml
```
apiVersion: v1
kind: Service
metadata:
  name: nginx-release-nginx
  labels:
    app.kubernetes.io/name: nginx
    app.kubernetes.io/instance: nginx
spec:
```



```
  ports:
    - port: 80
      targetPort: http
      protocol: TCP
      name: http
  selector:
    app: hello-world
```

```
  ports:
    - port: 80
      targetPort: http
      protocol: TCP
      name: http
  selector:
    app: hello-world
```

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

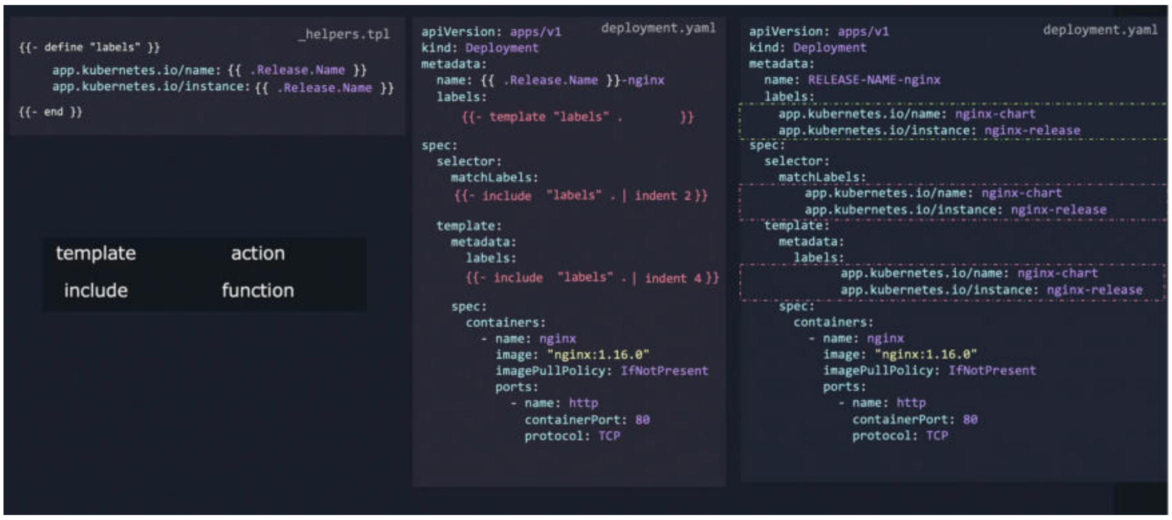


```
_helpers.tpl
{{- define "labels" }}
    app.kubernetes.io/name: {{ .Release.Name }}
    app.kubernetes.io/instance: {{ .Release.Name }}
{{- end }}
```

```
service.yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}-nginx
  labels:
     {{- template "labels" . }}
spec:
  ports:
    - port: 80
      targetPort: http
      protocol: TCP
      name: http
  selector:
    app: hello-world
```

```
service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-release-nginx
  labels:
    app.kubernetes.io/name: nginx
    app.kubernetes.io/instance: nginx
spec:
  ports:
    - port: 80
      targetPort: http
      protocol: TCP
      name: http
  selector:
    app: hello-world
```

However the spacing will give an issue as we might need extra indentation in another part of the file
- need to add " l indent 4 "
- but this don't work as template functions don't have the indentation function
- must use "include"



```
_helpers.tpl
{{- define "labels" }}
    app.kubernetes.io/name: {{ .Release.Name }}
    app.kubernetes.io/instance: {{ .Release.Name }}
{{- end }}
```

template	action
include	function

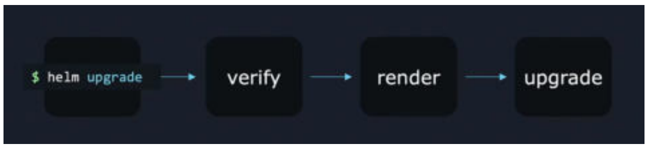
```
deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name }}-nginx
  labels:
     {{- template "labels" .        }}
spec:
  selector:
    matchLabels:
    {{- include  "labels" . | indent 2 }}
  template:
    metadata:
      labels:
      {{- include  "labels" . | indent 4 }}
    spec:
      containers:
        - name: nginx
          image: "nginx:1.16.0"
          imagePullPolicy: IfNotPresent
          ports:
            - name: http
              containerPort: 80
              protocol: TCP
```

```
deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: RELEASE-NAME-nginx
  labels:
    app.kubernetes.io/name: nginx-chart
    app.kubernetes.io/instance: nginx-release
spec:
  selector:
    matchLabels:
        app.kubernetes.io/name: nginx-chart
        app.kubernetes.io/instance: nginx-release
  template:
    metadata:
      labels:
            app.kubernetes.io/name: nginx-chart
            app.kubernetes.io/instance: nginx-release
    spec:
      containers:
        - name: nginx
          image: "nginx:1.16.0"
          imagePullPolicy: IfNotPresent
          ports:
            - name: http
              containerPort: 80
              protocol: TCP
```

Chart Hooks

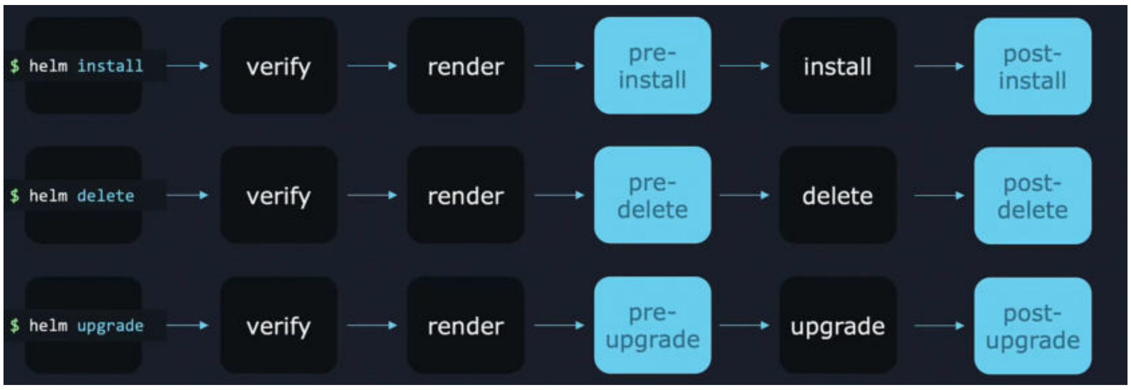
Useful in the following scenario
- whenever we do a "helm upgrade word-press bitnami/wordpress"
- we can trigger a database backup hook to backup the db before upgrade

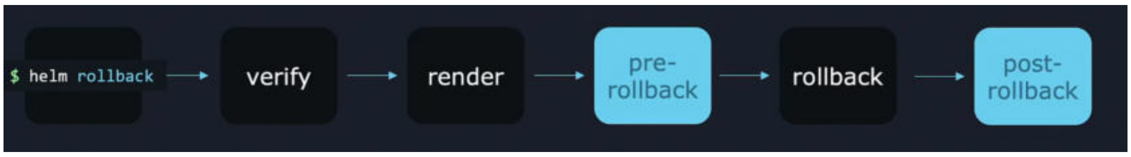
Normally when a user upgrades/install a helm file
- he will verify and render a file then install the file on the cluster



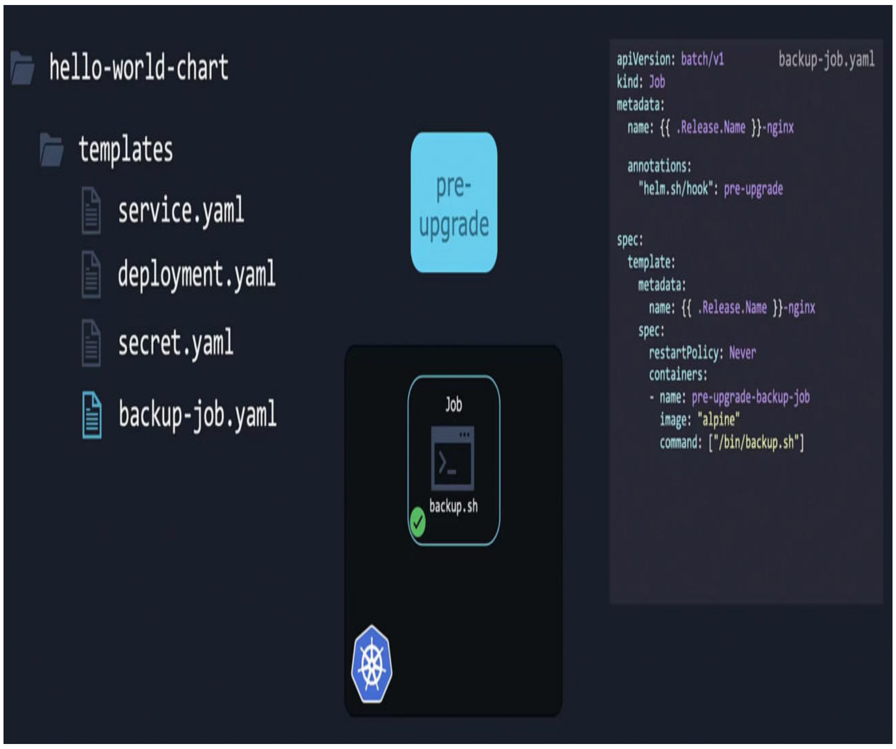
In our case, we need to define the pre-upgrade/post-upgrade hook before/after the upgrade

Many kinds of hook
- pre-install/post-install
- pre-delete/post-delete
- pre-upgrade/post-upgrade
- pre-rollback/post-rollback





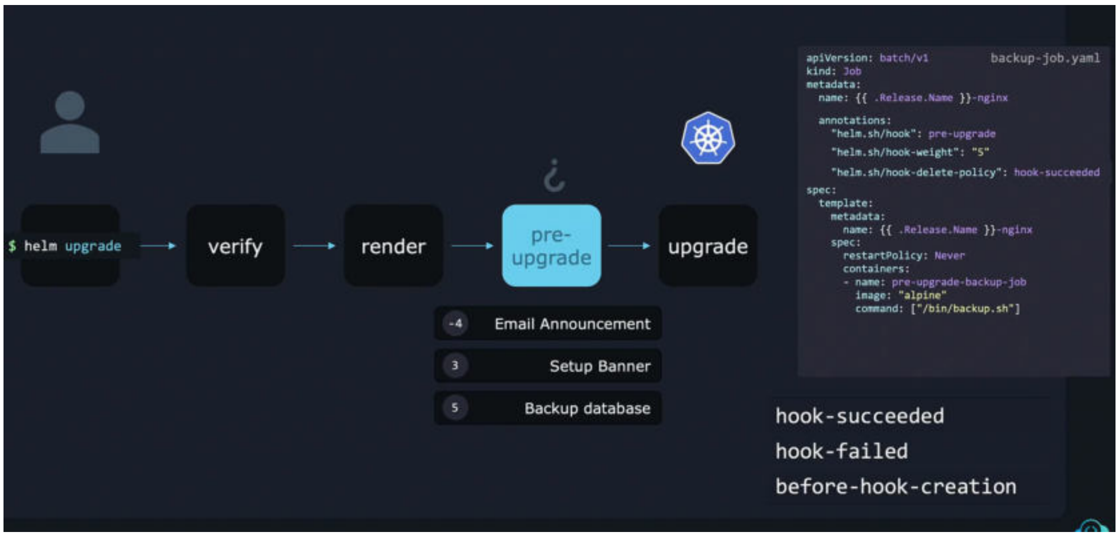
This hooks are ran as jobs - normally has a script that we run
- this jobs, we need to add the pre-upgrade annotation



```yaml
apiVersion: batch/v1          backup-job.yaml
kind: Job
metadata:
  name: {{ .Release.Name }}-nginx

  annotations:
    "helm.sh/hook": pre-upgrade

spec:
  template:
    metadata:
      name: {{ .Release.Name }}-nginx
    spec:
      restartPolicy: Never
      containers:
      - name: pre-upgrade-backup-job
        image: "alpine"
        command: ["/bin/backup.sh"]
```

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

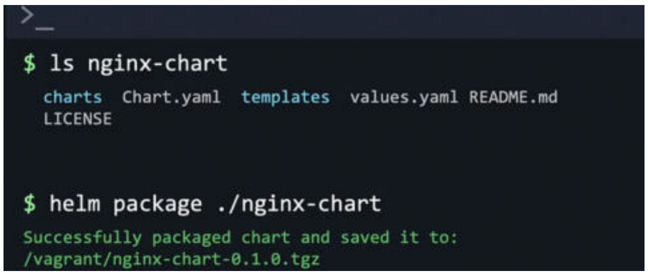


```yaml
apiVersion: batch/v1          backup-job.yaml
kind: Job
metadata:
  name: {{ .Release.Name }}-nginx

  annotations:
    "helm.sh/hook": pre-upgrade
    "helm.sh/hook-weight": "5"
    "helm.sh/hook-delete-policy": hook-succeeded
spec:
  template:
    metadata:
      name: {{ .Release.Name }}-nginx
    spec:
      restartPolicy: Never
      containers:
      - name: pre-upgrade-backup-job
        image: "alpine"
        command: ["/bin/backup.sh"]
```

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 ~/.gnupug/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

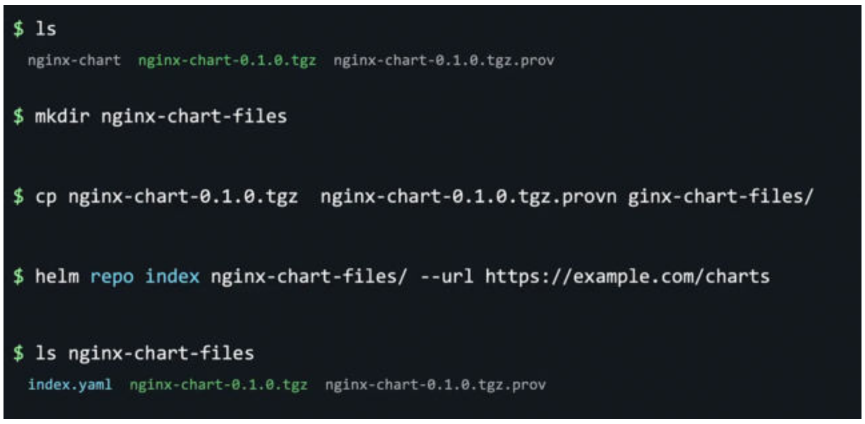
Verify signature
- helm verify ./nginx-chart-0.1.0.tgz

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

Uploadin g 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>