title: Helm_3 date: 2019-06-8 11:14:14 categories:

- K8S
- Helm tags:
- K8S
- Helm

HELM

部署

```
wget https://get.helm.sh/helm-v3.3.4-linux-amd64.tar.gz
tar xf helm-v3.3.4-linux-amd64.tar.gz
mv linux-amd64/helm /usr/local/bin
```

补全

```
source <(helm completion bash)
echo 'source <(helm completion bash)' >> .bashrc
```

常用仓库

- 微软仓库（http://mirror.azure.cn/kubernetes/charts/）这个仓库强烈推荐，基本上官网有的chart这里都有。
- 阿里云仓库（https://kubernetes.oss-cn-hangzhou.aliyuncs.com/charts ）
- 官方仓库（https://hub.kubeapps.com/charts/incubator）官方chart仓库，国内有点不好使。

添加仓库

```
#添加仓库
helm repo add stable http://mirror.azure.cn/kubernetes/charts/
```

删除仓库

```
helm repo remove stable
```

搜索chart

```
helm search repo nginx
```

chart下载到本地

```
helm pull google/nginx-ingress
```

自建chart

```
helm create mychart
```

chart打包

```
helm package mychart
```

查看chart信息

```
helm show chart stable/mysql
```

安装chart

```
helm install db stable/mysql
```

安装选项

- --values（或-f）：指定带有覆盖的YAML文件。这可以多次指定，最右边的文件优先
- --set：在命令行上指定替代values.yaml文件中的层级变量值。如果两者都用，--set优先级高
- -n 指定名称空间
- --dry-run
- --debug 调试参数

卸载

```
helm uninstall stable/mysql
```

升级

```
helm upgrade --set imageTag=1.17 web mychart
#或
helm upgrade -f values.yaml web mychart
```

回滚

历史版本

```
helm history web
```

回滚指定版本

```
helm rollback web 2
```

查看历史版本配置信息

```
helm get --revision 1 web
```

查看安装状态

```
helm status db
```

chart结构

```
tree
.
└── nginx
    ├── charts
    ├── Chart.yaml
    ├── templates
    │   ├── deployment.yaml
    │   ├── _helpers.tpl
    │   ├── hpa.yaml
    │   ├── ingress.yaml
    │   ├── NOTES.txt
    │   ├── serviceaccount.yaml
    │   ├── service.yaml
    │   └── tests
    │       └── test-connection.yaml
    └── values.yaml
```

内置对象

刚刚我们使用 `{{.Release.Name}}` 将 release 的名称插入到模板中。这里的 Release 就是 Helm 的内置对象，下面是一些常用的内置对象：

Chart.Name	chart 名称
Release.Name	release 名字
Release.Namespace	release 命名空间
Release.Service	release 服务的名称
Release.Revision	release 修订版本号，从1开始累加

模板常用函数

quote

- 自动为变量值添加双引号

```
app: {{ quote .Values.label.app }}  或  app: {{ .Values.label.app | quote  }}
#如label.app=123渲染后如下  helm install --dry-run web ../mychart/ 查看渲染结果
app: "123"
```

default

- 给变量设置默认值 当变量值为空时 默认值生效

```
- name: {{ default "nginx" .Values.name   }} 或 - name: {{ .Values.name | default "nginx" }}
```

其他函数

缩进：{{ .Values.resources | indent 12 }} 或 {{ .Values.resources | nindent 12 }} # nindent是数组内容换行 indent 不换行

大写：{{ upper .Values.resources }}

首字母大写：{{ title .Values.resources }}

条件判断

运算符判断

eq	**检测两个数是否相等**
ne	检测两个数是否不相等
gt	检测左边的数是否大于右边的
lt	检测左边的数是否小于右边的
ge	检测左边的数是否大于等于右边的
le	检测左边的数是否小于等于右边的

```
# cat values.yaml
devops: k8

# cat templates/deployment.yaml
...
  template:
    metadata:
      labels:
        app: nginx
        {{- if eq .Values.devops "k8s" }}
        devops: 123
        {{- else }}
        devops: 456
        {{- end }}1
#{{- if ...}} 中的- 是为了渲染后消除 判断语句带来的空行
```

真假判断

- 判断变量是否为真 如果为真执行

默认为假的情况

- 一个布尔类型的 `假`
- 一个数字 `零`
- 一个 `空` 的字符串
- 一个 `nil` （空或 `null`）
- 一个空的集合（ `map`、 `slice`、 `tuple`、 `dict`、 `array`）

```
...
    spec:
      containers:
      - image: nginx:1.16
        name: nginx
        {{- if .Values.resources }}
```

```
        resources:
{{ toYaml .Values.resources | indent 10 }}
        {{- end }}
```

判断一个空的数组.

```
# cat values.yaml
resources: {}
  # limits:
  #   cpu: 100m
  #   memory: 128Mi
  # requests:
  #   cpu: 100m
  #   memory: 128Mi
```

```
# cat templates/deploymemt.yaml
...
    spec:
      containers:
      - image: nginx:1.16
        name: nginx
        {{- if .Values.resources }}
        resources:
{{ toYaml .Values.resources | indent 10 }}
        {{- end }}
```

判断一个布尔值

```
# cat values.yaml
service:
  type: ClusterIP
  port: 80

ingress:
  enabled: true
  host: example.ctnrs.com
```

```
# cat templates/ingress.yaml
{{- if .Values.ingress.enabled -}}
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: {{ .Release.Name }}-ingress
spec:
  rules:
  - host: {{ .Values.ingress.host }}
    http:
      paths:
      - path: /
        backend:
          serviceName: {{ .Release.Name }}
          servicePort: {{ .Values.service.port }}
{{ end }}
```

with

- 控制变量作用域 比如{{- with .Values.nodeSelector }} 就只能只用.Values.nodeSelector 中的变量 "." 就等于.Values.nodeSelector
- 如果想使用其他变量 可以使用 $ 引用 如 {{ $.Release.Name }}

```
# cat values.yaml
...
replicas: 3
label:
  project: ms
  app: nginx
```

```
# cat templates/deployment.yaml
apiVersion: apps/v1
kind: Deployment
...
    spec:
      {{- with .Values.nodeSelector }}
      nodeSelector:
        team: {{ .team }}
        gpu: {{ .gpu }}
      {{- end }}
      containers:
      - image: nginx:1.16
        name: nginx
```

```
#引用方法2 toYaml 函数 默认顶格写 需要使用nindent函数缩进
apiVersion: apps/v1
kind: Deployment
...
    spec:
      {{- with .Values.nodeSelector }}
      nodeSelector:
        {{- toYaml . | nindent 8 }}
      {{- end }}
      containers:
      - image: nginx:1.16
        name: nginx
```

循环 range

循环列表

```
# cat values.yaml
test:
  - 1
  - 2
  - 3
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}
```

```
data:
  test: |
  {{- range .Values.test }}
    {{ . }}
  {{- end }}
```

循环变量

```
# cat ../values.yaml
env:
  NAME: "gateway"
  JAVA_OPTS: "-Xmx1G"
```

```
# cat deployment.yaml
...
        env:
        {{- range $k, $v := .Values.env }}
           - name: {{ $k }}
             value: {{ $v | quote }}
        {{- end }}
```

```
#渲染结果如下
    env:
       - name: JAVA_OPTS
         value: "-Xmx1G"
       - name: NAME
         value: "gateway"
```

命名模板

- 命名模板内容尽量顶格写 否则后续处理缩进问题会很头疼

template引用

```
# cat _helpers.tpl
{{- define "demo.fullname" -}}
{{- .Chart.Name -}}-{{ .Release.Name }}
{{- end -}}
```

```
# cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ template "demo.fullname" . }}
...
```

include引用

- 由于template 引用命名模板后不能被其他函数二次处理 比如缩进函数,为了解决此问题helm加入include来引用命名模板实现其他函数的二次处理

```
# cat _helpers.tpl
{{- define "demo.labels" -}}
app: {{ template "demo.fullname" . }}
chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
release: "{{ .Release.Name }}"
{{- end -}}
```

```
# cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "demo.fullname" . }}
  labels:
    {{- include "demo.labels" . | nindent 4 }}
...
```

harbor作为chart仓库

harbor启用chart功能

- **harbor需要大于等于 v1.6.0 版本**

```
#harbor 目录下
./install.sh --with-chartmuseum
```

helm启用push插件

```
helm plugin install https://github.com/chartmuseum/helm-push
```

添加仓库

```
helm repo add  --username admin --password Harbor12345 myrepo
http://harbor_url/chartrepo/library
#library 是项目仓库
#chartrepo 是固定的
```

推送

```
helm push mysql-1.4.0.tgz --username=admin --password=Harbor12345
http://harbor_url/chartrepo/library
```