# Helm包管理工具

# 前言

- 本章主要讲述什么是kubernetes包管理工具Helm，Chart包的文件组成结构，Chart包中各个文件的作用等。

# 目标

- 学完本课程后，您将能够：
  - 描述Helm包管理工具的作用
  - 区分Chart和Release的不同
  - 使用Helm命令行完成安装，查看等操作

# 目录

# Helm简介

- What is Helm?
  - Helm Charts是 Kubernetes 项目中的一个子项目，该子项目现在托管在github上（https://github.com/kubernetes/helm），目的是提供 Kubernetes 的包管理平台。Helm 能够帮你管理 Kubernetes 的应用集合。Helm Charts 能够帮你定义，安装，升级最复杂的 Kubernetes 应用集合。
  - Helm Charts 很容易创建，做版本化，共享和发布，最新版本的 Helm 由 CNCF 进行维护，目前在业界已经有大量的公司在使用 Helm，其中包括谷歌，微软，Bitnami 等大型企业。

# Helm简介

- What Problems did Helm solve?
  - 在 Kubernetes中部署一个可以使用的应用，需要涉及到很多的 Kubernetes 资源共同协作。比如安装一个 WordPress ，用到了一些 K8s 的一些资源对象，包括 Deployment 用于部署应用、Service 提供服务发现、Secret 配置 WordPress 的用户名和密码，可能还需要 pv 和 pvc 来提供持久化服务。并且 WordPress 数据是存储在 mariadb里面的，所以需要 mariadb 启动就绪后才能启动 WordPress。这些 k8s 资源过于分散，不方便管理，通过 kubectl 来管理它们将会十分麻烦。所以我们在 k8s 中可以使用helm部署一个应用来解决以下几个问题：
    - 如何统一管理、配置和更新这些分散的 k8s 的应用资源文件
    - 如何分发和复用一套应用模板
    - 将应用的一系列资源当做一个软件包管理

# Helm简介

- Why Teams Love Helm

Manage Complexity

Easy Updates

Simple Sharing

Rollbacks

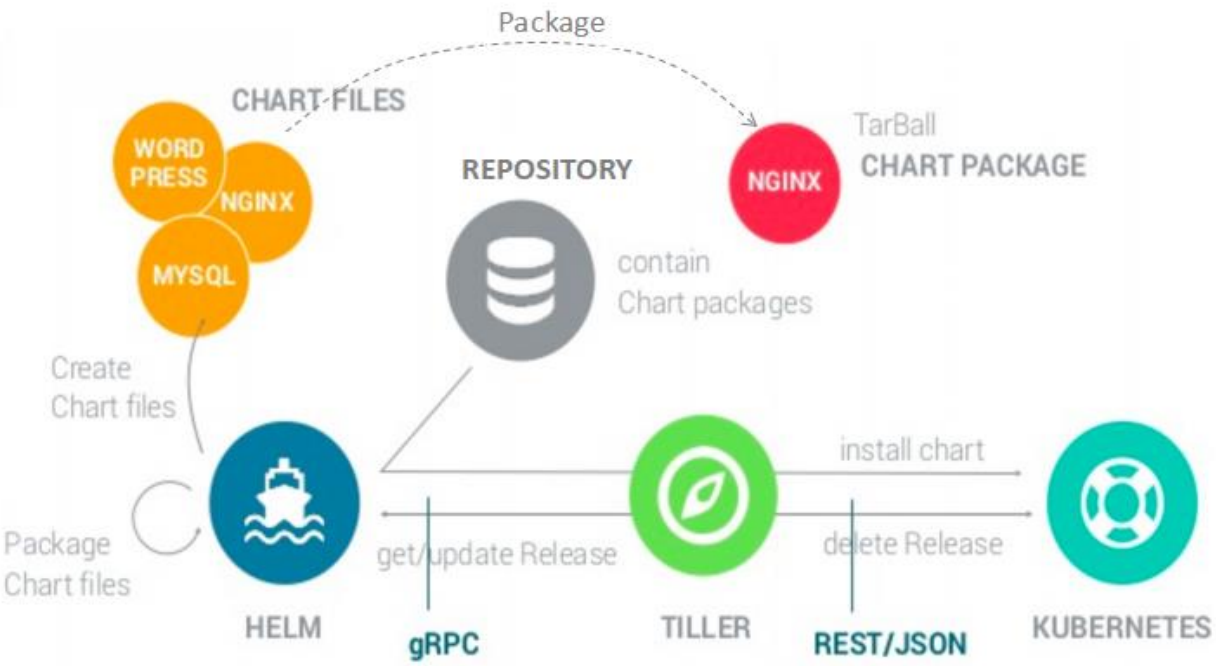
1、管理复杂的应用集合
Charts 能够描述最复杂的应用，提供可重复，幂等性的安装，以及提供统一的认证中心服务。
2、容易升级
为团队提供实时的镜像升级，以及自定义 webhook，解决镜像升级的痛点。
3、企业内部共享
Charts能够很容易的进行版本化，共享，在企业内部提供私有Helm 仓库服务，解决了从官方源拉镜像速度奇慢的痛点。
4、回滚
使用 Helm 可以方便的进行应用的回滚，回到之前的 Release 版本。

# Helm简介

- Helm相关概念
  - chart： Helm的打包格式叫做chart，所谓chart就是一系列文件, 它描述了一组相关的 k8s 集群资源；
  - Release：是chart的部署实例，一个chart在一个Kubernetes集群上可以有多个release；
  - Tiller Server：是 Helm 的服务端。Tiller 负责接收 Helm 的请求，与 k8s 的 apiserver 交互，根据chart 来生成一个 release 并管理 release ；
  - helm：是一个命令行工具，通过 gRPC 协议与 Tiller Server 进行交互，主要提供了增删查改 chart、release 和 repository 相关的功能；
  - Repoistory：Helm chart 的仓库，Helm 客户端通过 HTTP 协议来访问存储库中 chart 的索引文件和压缩包。

# Helm简介

- Helm架构
  - HelmClient是以Go编程语言编写的，使用gRPC协议套件与Tiller服务器进行交互
  - Tiller服务器也是在Go中编写的，它提供了一个gRPC服务器与客户端连接，它使用kubernet客户端库与kubernetes通信，目前该库使用REST+JSON；
  - Tiller服务器把信息存储在kubernetes中内置的ConfigMaps上，不需要自己的数据库。



# 目录






HUAWEI

# Helm基本操作

- Helm version查看当前版本

```
[root@master ~]# helm version
Client: &version.Version{SemVer:"v2.11.0", GitCommit:"2e55dbe1fdb5fdb96b75ff1
44a339489417b146b", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.11.0", GitCommit:"2e55dbe1fdb5fdb96b75ff1
44a339489417b146b", GitTreeState:"clean"}
```

- Helm repo list查看当前环境使用的仓库

```
[root@master ~]# helm repo list
NAME    URL
stable  https://kubernetes-charts.storage.googleapis.com
local   http://127.0.0.1:8879/charts
```

  - Helm 安装时已经默认配置好了两个仓库：stable 和 local。stable 是官方仓库，local 是用户存放自己开发的 chart 的本地仓库。

# Helm基本操作

- Helm search查看当前可安装的chart（从仓库中搜索）

```
[root@master ~]# helm search | more
NAME                                    CHART VERSION   APP VERSION             DESCRIPTION
stable/acs-engine-autoscaler            2.2.2           2.1.1                   DEPRECATED Scales wo
stable/aerospike                        0.2.5           v4.5.0.5                A Helm chart for Aer
stable/airflow                          2.8.2           1.10.2                  Airflow is a platfor
stable/ambassador                       2.4.1           0.61.0                  A Helm chart for Dat
stable/anchore-engine                   1.0.1           0.4.0                   Anchore container ar
stable/apm-server                       2.1.0           7.0.0                   The server receives
stable/ark                              4.2.2           0.10.2                  DEPRECATED A Helm ch
stable/artifactory                      7.3.1           6.1.0                   DEPRECATED Universal
stable/artifactory-ha                   0.4.1           6.2.0                   DEPRECATED Universal
stable/atlantis                         3.2.0           v0.7.1                  A Helm chart for Atl
stable/auditbeat                        1.1.0           6.7.0                   A lightweight shippe
stable/aws-cluster-autoscaler           0.3.3                                   Scales worker nodes
```

  - helm search 会显示 chart 位于哪个仓库，比如 local/****** 代表本地仓库和stable/******代表官方仓库。
  - 用户可以通过 helm repo add 添加更多的仓库

# Helm基本操作

- Helm search XXX 通过关键字搜索

```
[root@master ~]# helm search mysql
NAME                                    CHART VERSION   APP VERSION     DESCRIPTION
stable/mysql                            1.0.0           5.7.14          Fast, reliable, scalable, and easy
stable/mysqldump                        2.4.0           2.4.0           A Helm chart to help backup MySQL
stable/prometheus-mysql-exporter        0.3.2           v0.11.0         A Helm chart for prometheus mysql
stable/percona                          0.3.5           5.7.17          free, fully compatible, enhanced,
stable/percona-xtradb-cluster           0.7.0           5.7.19          free, fully compatible, enhanced,
stable/phpmyadmin                       2.2.0           4.8.5           phpMyAdmin is an mysql administrat
stable/gcloud-sqlproxy                  0.6.1           1.11            DEPRECATED Google Cloud SQL Proxy
stable/mariadb                          6.0.0           10.3.14         Fast, reliable, scalable, and easy
```

- Helm install XXX（仓库名）/XXX（软件名）

```
[root@master ~]# helm install stable/mysql
NAME:   coiling-angelfish
LAST DEPLOYED: Wed May 15 19:45:15 2019
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/Secret
NAME                    AGE
coiling-angelfish-mysql  1s

==> v1/ConfigMap
coiling-angelfish-mysql-test  1s
```

# Helm基本操作

- Helm install XXX（仓库名）/XXX（软件名）
  - 安装chart输出三部分
    - 部署基本信息：
      - Name：release的名字，可在安装的时候使用-n参数指定名字，这里因为在创建时未指定所以名字随机创建为original-hamster
      - LAST DEPOLYED：上一次部署时间
      - Namespace：release部署的namespace，默认为default，可以使用—namespace指定
      - Status：部署状态，这里是deployed，代表该chart已经部署到对应的namespace中
    - RESOURCES：
      - 该release中包含的资源有：Service、Deployment、Secret 、 PersistentVolumeClaim（即PVC）和configMap，其命名格式为 ReleaseName-ChartName。

# Helm基本操作

- Helm install XXX（仓库名）/XXX（软件名）
  - 安装chart输出三部分（分别对应图中1、2、3）
    - Notes：（介绍了数据库的连接方式）
      - 访问方式：包括3360端口访问，或者通过DNS域名访问：original-hamster-mysql.default.svc.cluster.local
      - 获取数据库root密码的方式
      - 集群内连接数据的步骤：
        - 1.运行一个ubuntu的pod作为客户端
        - 2.安装mysql专用客户端
        - 3.使用mysql命令行连接数据库，并输入密码
      - 集群外远程连接数据库的方式

# 目录

# Chart简介

- Helm的打包格式叫做chart，所谓chart就是一系列文件，它描述了一组相关的 kubernetes 集群资源。Chart中的文件安装特定的目录结构组织，以上文中Mysql Chart为例，一旦安装了某个Chart，则可以在~/.helm/cache/archive中看到对应的tar包，解压后使用tree命令目录结构如下：

```
[root@master archive]# tree mysql
mysql
├── Chart.yaml
├── README.md
├── templates
│   ├── configurationFiles-configmap.yaml
│   ├── deployment.yaml
│   ├── _helpers.tpl
│   ├── initializationFiles-configmap.yaml
│   ├── NOTES.txt
│   ├── pvc.yaml
│   ├── secrets.yaml
│   ├── svc.yaml
│   └── tests
│       ├── test-configmap.yaml
│       └── test.yaml
└── values.yaml

2 directories, 13 files
```

# Chart简介

```
[root@master archive]# tree -L 1 wordpress
wordpress
├── charts
├── Chart.yaml
├── README.md
├── requirements.lock
├── requirements.yaml
├── templates
└── values.yaml
```

- 目录名就是 Chart 的名字，主要包含如下内容：
  - Chart.yaml 包含Chart的基本信息，包括chart版本，名称（必选）等
  - README.md，可选文件，相当于该chart的使用文档
  - Requirements.yaml，chart可能依赖其它的chart，可以用该yaml指定
  - templates 目录下存放应用一系列 k8s 资源的 yaml 模板文件， Helm 会将 values.yaml 中的参数值注入到模板中生成标准的 yaml 配置文件

# Chart简介

```
[root@master archive]# tree -L 1 wordpress
wordpress
├── charts
├── Chart.yaml
├── README.md
├── requirements.lock
├── requirements.yaml
├── templates
└── values.yaml
```

- _helpers.tpl 此文件中定义一些可重用的模板片断，被其它地方调用
- NOTES.txt 介绍chart 部署后的帮助信息，如何使用chart等
- values.yaml 提供了这些配置参数的默认值， chart 支持在安装的时根据参数进行定制化配置
- charts/：目录，存放当前Charts依赖到的所有Charts文件
- templates/：目录，存放当前Charts用到的模板文件，可应用于Charts生成有效的Kubernetes清单文件

# 目录

# Chart.yaml文件

- Chart.yaml提供Charts相关的各种元数据，如名称、版本、关键词、维护者信息、使用的引擎模板等，它是一个Charts必备的核心文件，主要包含以下字段：

```
apiVersion: v1
appVersion: 5.7.14
description: Fast, reliable, scalable, and easy to use open-source relational database
  system.
engine: gotpl
home: https://www.mysql.com/
icon: https://www.mysql.com/common/logos/logo-mysql-170x115.png
keywords:
- mysql
- database
- sql
maintainers:
- email: o.with@sportradar.com
  name: olemarkus
- email: viglesias@google.com
  name: viglesiasce
name: mysql
sources:
- https://github.com/kubernetes/charts
- https://github.com/docker-library/mysql
version: 1.0.0
```

# Chart.yaml文件

- engine：模板引擎名称，默认为gotpl，即go模板
- icom：当前项目的图标指向的URL，可选
- appVersion：本项目用到的应用程序的版本号，可选
- tillerVersion：当前Charts依赖的Tiller版本号，可选

# requirements.yaml文件

- Helm中的一个Charts可能会依赖不止一个其它的Charts，这种依赖关系可经过requirements.yaml进行动态链接，该文件本质上是一个简单的依赖关系列表，主要包含以下字段：

```
dependencies:
- name: mariadb
  version: 5.x.x
  repository: https://kubernetes-charts.storage.googleapis.com/
  condition: mariadb.enabled
  tags:
    - wordpress-database
```

  - name：被依赖的Charts的名称
  - version：被依赖的Charts版本号

# requirements.yaml文件

- repository：被依赖的Charts所属的仓库及其URL；如果是非官方的仓库，则需要通过`helm repo add`命令将其添加进本地可用仓库
- alias：为被依赖的Charts创建一个别名
- tags：默认情况下所有的Charts都会被装载，但是若定义了tags，则仅装载那些匹配到的Charts
- name：被依赖的Charts的名称
- condition：类似于tags字段，但需要通过自定义条件指明需要装载的charts

# templates目录

- Helm Charts模板遵循Go模板语言格式，所有的模板文件都存储于templates目录中，在当前Charts被Helm引用时，此目录中的所有模板文件都会传递给模板引擎进行处理。
- 各类Kubernetes资源的配置模板都放置在这里。模板文件中用到的值（value）有如下几种提供方式：
  - 通过Charts的values.yaml文件提供，通常为默认值
  - 在运行“helm install”命令时通过--values参数传递包含所需要的自定义值的YAML文件，此处传递的值会覆盖默认值
  - 通过 --set 直接传入参数值

# Chart模板的使用

- 查看template目录下的secretes.yaml文件：

```
metadata:
  name: {{ template "mysql.fullname" . }}
  labels:
    app: {{ template "mysql.fullname" . }}
    chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
    release: "{{ .Release.Name }}"
    heritage: "{{ .Release.Service }}"
```

采用key-value格式，定义关键字的属性值：

```
name: {{ template "mysql.fullname" . }}
#该格式定义了secret的name（可自行查看pvc.yaml,svc.yaml,develoyment.yaml）
#template的功能是导入一个命名模板，导入了引号的内容，即mysql.fullname子模板
#mysql.fullname子模板在通用模板_helpers.tpl 被定义
```

# Chart模板的使用

- 查看template目录下的secretes.yaml文件：

_helpers.tpl中对mysql.fullname子模板的定义如下：

```
{{- define "mysql.fullname" -}}
{{- if .Values.fullnameOverride -}}
{{- .Values.fullnameOverride | trunc 63 | trimSuffix "-" -}}
{{- else -}}
{{- $name := default .Chart.Name .Values.nameOverride -}}
{{- if contains $name .Release.Name -}}
{{- printf .Release.Name | trunc 63 | trimSuffix "-" -}}
{{- else -}}
{{- printf "%s-%s" .Release.Name $name | trunc 63 | trimSuffix "-" -}}
{{- end -}}
```

# Chart模板的使用

- 查看template目录下的secretes.yaml文件：

```
labels:
    app: {{ template "mysql.fullname" . }}
    chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
    release: "{{ .Release.Name }}"
    heritage: "{{ .Release.Service }}"
```

  - 不管是chart或者是release都是对象，则根据相应对象的属性值可以得到

```
app: {{ template "mysql.fullname" . }} = coiling-angelfish-mysql

chart: {{ .Chart.Name }}-{{ .Chart.Version }} = mysql-5.7.14

release: {{ .Release.Name }} = coiling-angelfish

heritage: {{ .Release.Service }} = Tiller
```

# Chart模板的使用

- 查看template目录下的secretes.yaml文件：

```
data:
  {{ if .Values.mysqlRootPassword }}
  mysql-root-password:  {{ .Values.mysqlRootPassword | b64enc | quote }}
  {{ else }}
  mysql-root-password: {{ randAlphaNum 10 | b64enc | quote }}
  {{ end }}
  {{ if .Values.mysqlPassword }}
  mysql-password:  {{ .Values.mysqlPassword | b64enc | quote }}
  {{ else }}
  mysql-password: {{ randAlphaNum 10 | b64enc | quote }}
  {{ end }}
```

  - 使用了 if-else 的流控制，其逻辑为：
    - 如果 .Values.mysqlPassword 有值，则对其进行 base64 编码；否则随机生成一个 10 位的字符串并编码。
    - 这里的Values也是定义的对象，代表values.yaml文件， Values.mysqlRootPassword则代表values.yaml文件中定义的mysqlRootPassword参数

# Helm部署mysql应用实例

- 使用helm search mysql找到仓库中关键字包含mysql的Chart

```
[root@master /]# helm search mysql
NAME                                    CHART VERSION   APP VERSION
stable/mysql                            1.0.0           5.7.14
ce rel...
stable/mysqldump                        2.4.0           2.4.0
ysqldump
stable/prometheus-mysql-exporter        0.3.2           v0.11.0
udsqlp...
stable/percona                          0.3.5           5.7.17
```

- 使用helm inspect values查看chart的安装和使用方法

```
[root@master /]# helm inspect values stable/mysql
## mysql image version
## ref: https://hub.docker.com/r/library/mysql/tags/
##
image: "mysql"
imageTag: "5.7.14"

busybox:
  image: "busybox"
  tag: "1.29.3"
```

# Helm部署mysql应用实例

- 阅读这部分内容（输出的即Chart包中values.yaml的内容），在该实验中需要关注存储部分的注释

```
## Persist data to a persistent volume
persistence:
  enabled: true
  ## database data Persistent Volume Storage Class
  ## If defined, storageClassName: <storageClass>
  ## If set to "-", storageClassName: "", which disables dynamic provisioning
  ## If undefined (the default) or set to null, no storageClassName spec is
  ##   set, choosing the default provisioner.  (gp2 on AWS, standard on
  ##   GKE, AWS & OpenStack)
  ##
  # storageClass: "-"
  accessMode: ReadWriteOnce
  size: 8Gi
  annotations: {}
```

- 可以看到该chart安装需要：
  - 8G大小PersistentVolume
  - accessMode设置为ReadWriteOnce

# Helm部署mysql应用实例

- 根据要求编写好对应的PV文件，内容如下（该实验环境使用的是NFS PV）

```
[root@master pv_pvc]# cat nfs-pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-pv
spec:
  capacity:
    storage: 8Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
#  storageClassName: default
  nfs:
    path: /nfs
    server: 192.168.137.11
```

- 使用kubectl get pv查看状态为available

```
[root@master pv_pvc]# kubectl get pv
NAME       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
mysql-pv   8Gi        RWO            Recycle          Available                                   6s
```

# Helm部署mysql应用实例

- 使用helm install进行安装，这里我们使用--set传入参数值

```
[root@master pv_pvc]# helm install stable/mysql --set mysqlRootPassword=test123 -n test
NAME:   test
LAST DEPLOYED: Tue Jun 25 14:18:46 2019
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/Service
NAME        AGE
test-mysql  1s

==> v1beta1/Deployment
test-mysql  1s

==> v1/Pod(related)
```

- 通过helm status和helm list查看chart的部署状态

```
[root@master ~]# helm list
NAME    REVISION        UPDATED                         STATUS          CHART           APP VERSION
test    1               Tue Jun 25 14:18:46 2019        DEPLOYED        mysql-1.0.0     5.7.14
```

# Helm部署mysql应用实例

- 验证登录mysql，具体登录方法通过helm status查看

```
To connect to your database:

1. Run an Ubuntu pod that you can use as a client:

    kubectl run -i --tty ubuntu --image=ubuntu:16.04 --restart=Never -- bash -il

2. Install the mysql client:

    $ apt-get update && apt-get install mysql-client -y

3. Connect using the mysql cli, then provide your password:
    $ mysql -h test-mysql -p

To connect to your database directly from outside the K8s cluster:
    MYSQL_HOST=127.0.0.1
    MYSQL_PORT=3306

    # Execute the following command to route the connection:
    kubectl port-forward svc/test-mysql 3306

    mysql -h ${MYSQL_HOST} -P${MYSQL_PORT} -u root -p${MYSQL_ROOT_PASSWORD}
```

# Helm部署mysql应用实例

- 验证登录mysql，具体登录方法通过helm status查看

```
[root@master ~]# kubectl port-forward svc/test-mysql 3306
Forwarding from 127.0.0.1:3306 -> 3306
Forwarding from [::1]:3306 -> 3306
Handling connection for 3306
```

```
[root@master pv_pvc]# mysql -h 127.0.0.1 -P3306 -u root -ptest123
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 171
Server version: 5.7.14 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

# 实验&实训任务

- 实验任务
  - 请按照手册2.16章节完成Helm相关实验，包括：
    - 自定义创建Chart
    - 自定义应用Push到Local Registry
    - Helm升级回退应用
    - 搭建本地私有仓库
- 实训任务
  - 请灵活使用本章课程及使用手册中学到的知识，按照2.16.5章节完成实训任务

# 本章总结

- 本章节介绍了kubernetes包管理工具，包括如下：
  - Helm的作用
  - Chart包文件结构
  - Chart包模板文件的使用
  - Helm命令工具的简单操作


谢 谢
www.huawei.com