# 15Helm 专题（三） 使用 Helm

## Helm 专题（三） 使用 Helm

本文介绍了使用 Helm 来管理 Kubernetes 集群上的软件包的基础知识。在这之前，假定您已经 安装( `https://helm.sh/zh/docs/intro/install` )了 Helm 客户端。

Helm 安装 **charts** 到 Kubernetes 集群中，每次安装都会创建一个新的 *release* 。你可以在 Helm 的 chart *repositories* 中寻找新的 chart。

## 'helm search'：查找 Charts

Helm 自带一个强大的搜索命令，可以用来从两种来源中进行搜索：

- `helm search hub` 从 Artifact Hub( `https://artifacthub.io/` ) 中查找并列出 helm charts。Artifact Hub中存放了大量不同的仓库。
- `helm search repo` 从你添加（使用 `helm repo add` ）到本地 helm 客户端中的仓库中进行查找。该命令基于本地数据进行搜索，无需连接互联网。

## 'helm install'：安装一个 helm 包

使用 `helm install` 命令来安装一个新的 helm 包。最简单的使用方法只需要传入两个参数：你命名的release名字和你想安装的chart的名称。

```
$ helm install happy-panda bitnami/wordpress
NAME: happy-panda
LAST DEPLOYED: Tue Jan 26 10:27:17 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
** Please be patient while the chart is being deployed **

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

    happy-panda-wordpress.default.svc.cluster.local (port 80)

To access your WordPress site from outside the cluster follow the steps below:

1. Get the WordPress URL by running these commands:

  NOTE: It may take a few minutes for the LoadBalancer IP to be available.
        Watch the status with: 'kubectl get svc --namespace default -w happy-panda-wordpress'

   export SERVICE_IP=$(kubectl get svc --namespace default happy-panda-wordpress --template "{{ range (ind
   echo "WordPress URL: http://$SERVICE_IP/"
   echo "WordPress Admin URL: http://$SERVICE_IP/admin"

2. Open a browser and access WordPress using the obtained URL.

3. Login with the following credentials below to see your blog:

  echo Username: user
  echo Password: $(kubectl get secret --namespace default happy-panda-wordpress -o jsonpath="{.data.wordpr
```

现在 `wordpress` chart 已经安装。注意安装chart时创建了一个新的 *release* 对象。上述发布被命名为 `happy-panda` 。（如果想让Helm生成一个名称，删除发布名称并使用 `-generate-name` 。）

在安装过程中， `helm` 客户端会打印一些有用的信息，其中包括：哪些资源已经被创建，release当前的状态，以及你是否还需要执行额外的配置步骤。

Helm按照以下顺序安装资源：

- Namespace
- NetworkPolicy
- ResourceQuota
- LimitRange
- PodSecurityPolicy
- PodDisruptionBudget
- ServiceAccount
- Secret
- SecretList

- ConfigMap
- StorageClass
- PersistentVolume
- PersistentVolumeClaim
- CustomResourceDefinition
- ClusterRole
- ClusterRoleList
- ClusterRoleBinding
- ClusterRoleBindingList
- Role
- RoleList
- RoleBinding
- RoleBindingList
- Service
- DaemonSet
- Pod
- ReplicationController
- ReplicaSet
- Deployment
- HorizontalPodAutoscaler
- StatefulSet
- Job
- CronJob
- Ingress
- APIService

Helm 客户端不会等到所有资源都运行才退出。许多 charts 需要大小超过 600M 的 Docker 镜像，可能需要很长时间才能安装到集群中。

你可以使用 `helm status` 来追踪 release 的状态，或是重新读取配置信息：

```
$ helm status happy-panda
NAME: happy-panda
LAST DEPLOYED: Tue Jan 26 10:27:17 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
** Please be patient while the chart is being deployed **

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

    happy-panda-wordpress.default.svc.cluster.local (port 80)

To access your WordPress site from outside the cluster follow the steps below:

1. Get the WordPress URL by running these commands:

  NOTE: It may take a few minutes for the LoadBalancer IP to be available.
        Watch the status with: 'kubectl get svc --namespace default -w happy-panda-wordpress'

   export SERVICE_IP=$(kubectl get svc --namespace default happy-panda-wordpress --template "{{ range (ind
   echo "WordPress URL: http://$SERVICE_IP/"
   echo "WordPress Admin URL: http://$SERVICE_IP/admin"

2. Open a browser and access WordPress using the obtained URL.

3. Login with the following credentials below to see your blog:

  echo Username: user
  echo Password: $(kubectl get secret --namespace default happy-panda-wordpress -o jsonpath="{.data.wordpr
```

上述信息展示了 release 的当前状态。

## 安装前自定义 chart

上述安装方式只会使用 chart 的默认配置选项。很多时候，我们需要自定义 chart 来指定我们想要的配置。

使用 `helm show values` 可以查看 chart 中的可配置选项：

```
$ helm show values bitnami/wordpress
## Global Docker image parameters
## Please, note that this will override the image parameters, including dependencies, configured to use th
## Current available global Docker image parameters: imageRegistry and imagePullSecrets
##
# global:
#   imageRegistry: myRegistryName
#   imagePullSecrets:
#     - myRegistryKeySecretName
#   storageClass: myStorageClass

## Bitnami WordPress image version
## ref: https://hub.docker.com/r/bitnami/wordpress/tags/
##
image:
  registry: docker.io
  repository: bitnami/wordpress
  tag: 5.6.0-debian-10-r35
  [..]
```

然后，你可以使用 YAML 格式的文件覆盖上述任意配置项，并在安装过程中使用该文件。

```
$ echo '{mariadb.auth.database: user0db, mariadb.auth.username: user0}' > values.yaml
$ helm install -f values.yaml bitnami/wordpress --generate-name
```

上述命令将为 MariaDB 创建一个名称为 `user0` 的默认用户，并且授予该用户访问新建的 `user0db` 数据库的权限。chart 中的其他默认配置保持不变。

安装过程中有两种方式传递配置数据：

- `--values`(或 `f` )：使用 YAML 文件覆盖配置。可以指定多次，优先使用最右边的文件。
- `--set`：通过命令行的方式对指定项进行覆盖。

如果同时使用两种方式，则 `--set` 中的值会被合并到 `--values` 中，但是 `--set` 中的值优先级更高。在 `--set` 中覆盖的内容会被被保存在 ConfigMap 中。可以通过 `helm get values <release-name>` 来查看指定 release 中 `--set` 设置的值。也可以通过运行 `helm upgrade` 并指定 `--reset-values` 字段来清除 `--set` 中设置的值。

## `--set` 的格式和限制

`--set` 选项使用0或多个 name/value 对。最简单的用法类似于：`--set name=value`，等价于如下 YAML 格式：

```
name: value
```

多个值使用逗号分割，因此 `-set a=b,c=d` 的 YAML 表示是：

```
a: b
c: d
```

支持更复杂的表达式。例如，`--set outer.inner=value` 被转换成了：

```
outer:
  inner: value
```

列表使用花括号（`{}`）来表示。例如，`--set name={a, b, c}` 被转换成了：

```
name:
  - a
  - b
  - c
```

从 2.5.0 版本开始，可以使用数组下标的语法来访问列表中的元素。例如 `--set servers[0].port=80` 就变成了：

```
servers:
  - port: 80
```

多个值也可以通过这种方式来设置。`--set servers[0].port=80,servers[0].host=example` 变成了：

```
servers:
  - port: 80
    host: example
```

如果需要在 `-set` 中使用特殊字符，你可以使用反斜线来进行转义；`--set name=value1\,value2` 就变成了：

```
name: "value1,value2"
```

类似的，你也可以转义点序列（英文句号）。这可能会在 chart 使用 `toYaml` 函数来解析 annotations，labels，和 node selectors 时派上用场。`--set nodeSelector."kubernetes\.io/role"=master` 语法就变成了：

```
nodeSelector:
  kubernetes.io/role: master
```

深层嵌套的数据结构可能会很难用 `-set` 表达。我们希望 Chart 的设计者们在设计 `values.yaml` 文件的格式时，考虑到 `--set` 的使用。

## 更多安装方法

- chart 的仓库（如上所述）
- 本地 chart 压缩包（helm install foo foo-0.1.1.tgz）
- 解压后的 chart 目录（helm install foo path/to/foo）
- 完整的 URL（helm install foo https://example.com/charts/foo-1.2.3.tgz）

## 'helm upgrade' 和 'helm rollback'：升级 release 和失败时恢复

当你想升级到 chart 的新版本，或是修改 release 的配置，你可以使用 `helm upgrade` 命令。

一次升级操作会使用已有的 release 并根据你提供的信息对其进行升级。由于 Kubernetes 的 chart 可能会很大而且很复杂，Helm 会尝试执行最小侵入式升级。即它只会更新自上次发布以来发生了更改的内容。

```
$ helm upgrade -f panda.yaml happy-panda bitnami/wordpress
```

在上面的例子中，`happy-panda` 这个 release 使用相同的 chart 进行升级，但是使用了一个新的 YAML 文件：

```
mariadb.auth.username: user1
```

我们可以使用 `helm get values` 命令来看看配置值是否真的生效了：

```
$ helm get values happy-panda
mariadb:
  auth:
    username: user1
```

`helm get` 是一个查看集群中 release 的有用工具。正如我们上面所看到的，`panda.yaml` 中的新值已经被部署到集群中了。

现在，假如在一次发布过程中，发生了不符合预期的事情，也很容易通过 `helm rollback [RELEASE] [REVISION]` 命令回滚到之前的发布版本。

```
$ helm rollback happy-panda 1
```

上面这条命令将我们的 `happy-panda` 回滚到了它最初的版本。release 版本其实是一个增量修订（revision）。每当发生了一次安装、升级或回滚操作，revision 的值就会加1。第一次 revision 的值永远是1。我们可以使用 `helm history [RELEASE]` 命令来查看一个特定 release 的修订版本号。

## 安装、升级、回滚时的有用选项

你还可以指定一些其他有用的选项来自定义 Helm 在安装、升级、回滚期间的行为。请注意这并不是 cli 参数的完整列表。要查看所有参数的说明，请执行 `helm <command> --help` 命令。

- `--timeout`：一个 Go duration 类型的值，用来表示等待 Kubernetes 命令完成的超时时间，默认值为 `5m0s`。
- `--wait`：表示必须要等到所有的 Pods 都处于 ready 状态，PVC 都被绑定，Deployments 都至少拥有最小 ready 状态 Pods 个数（`Desired` 减去 `maxUnavailable`），并且 Services 都具有 IP 地址（如果是 `LoadBalancer`，则为 Ingress），才会标记该 release 为成功。最长等待时间由 `--timeout` 值指定。如果达到超时时间，release 将被标记为 `FAILED`。注意：当 Deployment 的 `replicas` 被设置为1，但其滚动升级策略中的 `maxUnavailable` 没有被设置为0时，`--wait` 将返回就绪，因为已经满足了最小 ready Pod 数。
- `--no-hooks`：不运行当前命令的钩子。

- `--recreate-pods` （仅适用于 `upgrade` 和 `rollback` ）：这个参数会导致重建所有的 Pod（deployment中的Pod 除外）。（在 Helm 3 中已被废弃）

## 'helm uninstall'：卸载 release

使用 `helm uninstall` 命令从集群中卸载一个 release：

```
$ helm uninstall happy-panda
```

该命令将从集群中移除指定 release。你可以通过 `helm list` 命令看到当前部署的所有 release：

```
$ helm list NAME VERSION UPDATED STATUS CHART inky-cat 1 Wed Sep 28 12:59:46 2016 DEPLOYED alpine-0.1.0
```

在上一个 Helm 版本中，当一个 release 被删除，会保留一条删除记录。而在 Helm 3 中，删除也会移除 release 的记录。如果你想保留删除记录，使用 `helm uninstall --keep-history` 。使用 `helm list --uninstalled` 只会展示使用了 `--keep-history` 删除的 release。

`helm list --all` 会展示 Helm 保留的所有 release 记录，包括失败或删除的条目（指定了 `--keep-history` ）：

```
$ helm list --all NAME VERSION UPDATED STATUS CHART happy-panda 2 Wed Sep 28 12:47:54 2016 UNINSTALLED wordpress-10.4.5.6.0 inky-cat 1 Wed Sep 28 12:59:46 2016 DEPLOYED alpine-0.1.0 kindred-angelf 2 Tue Sep 27 16:16:10 2016 UNINSTALLED alpine-0.1.0
```

注意，因为现在默认会删除 release，所以你不再能够回滚一个已经被卸载的资源了。

## 'helm repo'：使用仓库

Helm 3 不再附带一个默认的 chart 仓库。 `helm repo` 提供了一组命令用于添加、列出和移除仓库。

使用 `helm repo list` 来查看配置的仓库：

```
$ helm repo list NAME URL stable https://charts.helm.sh/stable mumoshu https://mumoshu.github.io/charts
```

使用 `helm repo add` 来添加新的仓库：

```
$ helm repo add dev https://example.com/dev-charts
```

因为 chart 仓库经常在变化，在任何时候你都可以通过执行 `helm repo update` 命令来确保你的 Helm 客户端是最新的。

使用 `helm repo remove` 命令来移除仓库。

## 命令总结

1. 查找 Charts

- `helm search hub xxxx` —从 Artifact Hub 中查找 charts
- `helm search repo xxxx` —从你添加到本地仓库中进行查找，基于本地数据搜索，无需互联网

2. 安装 Chart

- `helm install <release name> <chart name>` —自己手动命名 release
- `helm install <chart name> --generate-name` —自动生成名称
- `helm status <release name>` —追踪 release 状态
- `helm install foo foo-0.1.1.tgz` —本次chart压缩包
- `helm install foo path/to/foo` —解压后chart 目录
- `helm install foo [https://example.com/charts/foo-1.2.3.tgz](https://example.com/charts/foo-1.2.3.tgz)` —完整url

3. 安装前自定义 chart

- `--set name=value`
- `--set a=b,c=d`
- `--set outer.inner=value`
- `--set name={a, b, c}`
- `--set servers[0].port=80`
- `--set name=value1\,value2`
- `--set nodeSelector."kubernetes\.io/role"=master`
- `helm show values <chart name>` —查看 chart 中 可配置的选项
- `helm install -f values.yaml <release name>` —通过yaml 指定配置 执行安装
- 通过命令行覆盖配置:
- `helm get values <release name>` —查看配置值

4. 升级

- `helm upgrade -f xxx.yaml <release name> <chart name>`

5. 回滚

- `helm rollback <release><revision>` —回滚到之前版本

6. 卸载

- `helm uninstall <release name>` —卸载release
- `helm list <-A 所有命名空间><-n 命名空间>` —查询部署的 release
- `helm uninstall --keep-history` 保留删除记录
- `helm list --uninstalled` 只会查看保留删除的记录

7. 仓库相关

- `helm repo list` 查看已配置的仓库
- `helm repo add <name> <url>` 添加新仓库
- `helm repo remove <name>` 删除仓库

欢迎关注我的公众号“**云原生拓展**”，原创技术文章第一时间推送。