版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

Kubernetes Helm v3 包管理工具

前言：

课程名称：Kubernetes Helm v3 包管理工具

实验环境：

本章节 Kubernetes 集群环境如下：

角色	IP	主机名	组件	硬件
控制节点	192.168.128.11	k8s-master01	apiserver controller-manager scheduler etcd containerd	CPU：4vCPU 硬盘：100G 内存：4GB 开启虚拟化
工作节点	192.168.128.21	k8s-node01	kubelet kube-proxy containerd calico coredns	CPU：6vCPU 硬盘：100G 内存：6GB 开启虚拟化

张岩峰老师微信，加我微信，邀请你加入 VIP 交流答疑群：

微信号：ZhangYanFeng0429

二维码：

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

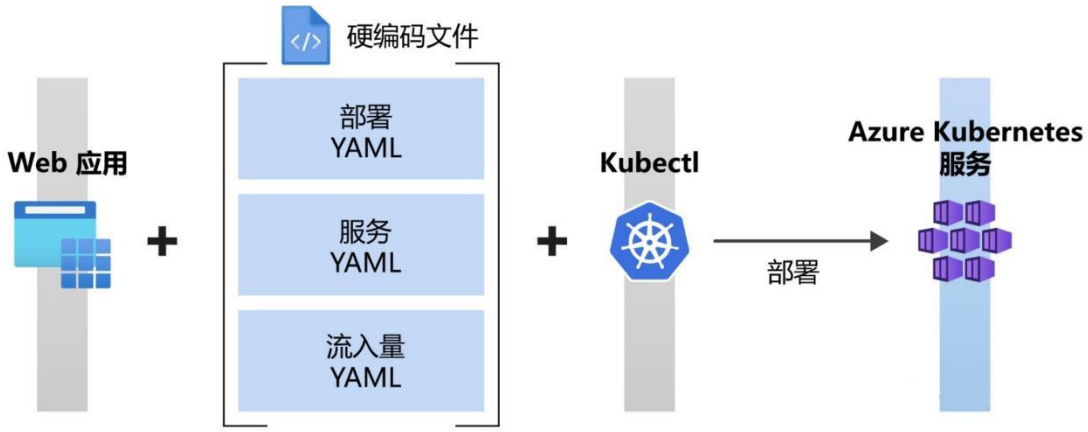
版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

1、初识 Helm

1.1、Helm 介绍

官网文档站点：https://v3.helm.sh/zh/docs/
helm 官方的 chart 站点：https://hub.kubeapps.com/

传统的 K8s 部署方式：



随着引用增多，需要维护大量的 yaml 文件。不能根据一套 yaml 文件来创建多个环境。当环境发生变化，需要手动修改 yaml 文件，就会比较费劲。

什么是 Helm?

Helm 是 kubernetes 的包管理工具，相当于 linux 环境下的 yum/apt-get 命令。

Helm 的目标一直是让“从零到 Kubernetes”变得轻松。无论是运维、开发人员、经验丰富的 DevOps 工程师，还是刚刚入门的学生，Helm 的目标是让大家在两分钟内就可以在 Kubernetes 上安装应用程序。

1.2、Helm 可以解决那些问题？

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

Helm 可以解决以下问题：

1、Helm 可以管理和发布 Kubernetes 应用，将 Kubernetes 资源（如 Deployment、Service、Ingress 等）封装到一个 Chart 中，通过 Chart 仓库进行存储和分享，简化 Kubernetes 部署应用的版本控制、打包、发布、删除、更新等操作。

2、Helm 使得发布可配置，支持发布应用配置的版本管理，简化了 Kubernetes 部署应用的版本控制、打包、发布、删除、更新等操作。

总之，Helm 是一个用于在 Kubernetes 上部署和管理应用的高层次工具，能够方便地管理 Kubernetes 应用资源，简化操作的复杂性。

1.3、Helm v3 版本变化

Helm V2 核心组件：

Helm V2 的核心组件包括两个：Helm 客户端和 Tiller 服务器。

- Helm 客户端

Helm 客户端是用于与 Kubernetes 进行交互的命令行工具。它能够执行 Helm 命令，这些命令会与 Kubernetes API Server 进行交互，用于创建、查询、删除、更新 Kubernetes 资源。Helm 客户端还可以与 Tiller 服务器进行交互，执行定义在 Chart 中的模板化配置命令。

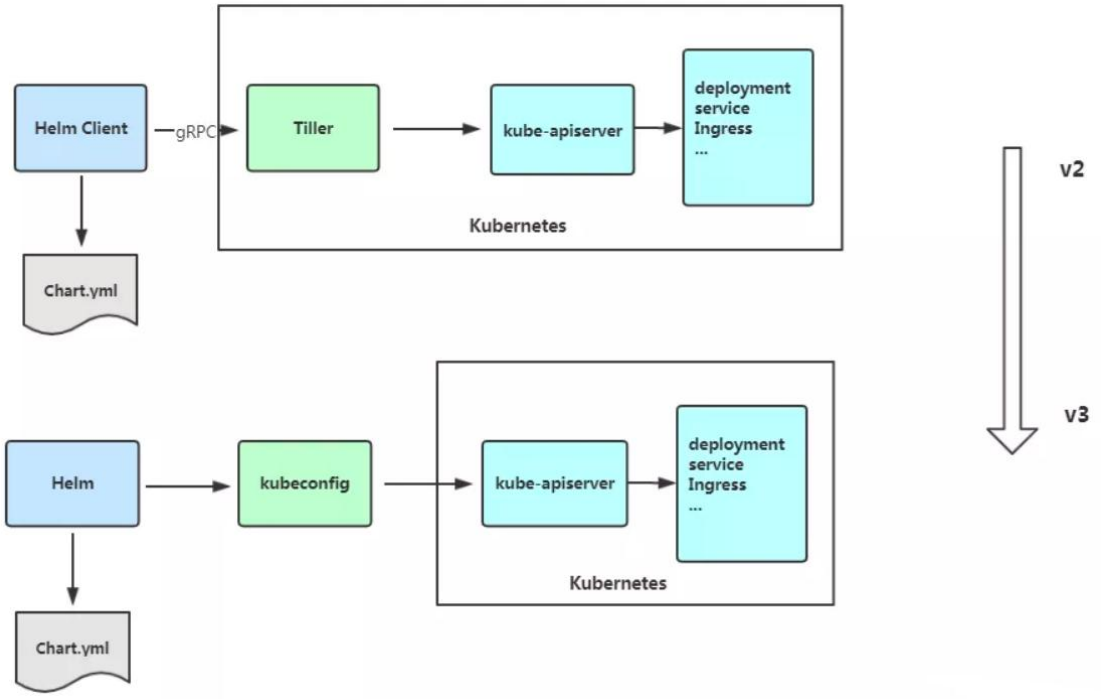
- Tiller 服务器

Tiller 服务器是一个运行在 Kubernetes 集群中的服务，它负责处理 Helm 命令的执行。Helm 客户端将 Helm 命令发送给 Tiller 服务器，Tiller 服务器则会执行这些命令。Tiller 服务器能够与 Kubernetes API Server 直接交互，执行各种 Kubernetes 资源操作。由于 Helm 的设计，使得 Tiller 服务器具有一些安全性优势，例如支持定义策略来限制对 Kubernetes 资源的访问。

v2 与 v3 对比图：

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。



Helm v3 的核心组件包括以下三个：

- Helm 客户端（helm client）

  Helm 客户端是用户用于与 Kubernetes 进行交互的命令行工具。它负责处理 Helm 命令的发送和与 Kubernetes API Server 进行通信，用于创建、查询、删除、更新 Kubernetes 资源。在 Helm v3 中，Helm 客户端与 Kubernetes API Server 直接通信，不再需要经过 Tiller 服务器。

- Tillerless 模式（Tillerless Mode）

  在 Helm v3 中，引入了 Tillerless 模式，使得 Helm 可以完全摆脱对 Tiller 服务器的依赖。在 Tillerless 模式下，Helm 命令可以直接在 Helm 客户端上执行，无需与 Tiller 服务器进行交互。这简化了 Helm 的使用和部署过程。

- Kubernetes 资源（Kubernetes resources）：

  Helm v3 将 Kubernetes 的资源封装到一个 Chart 中，Chart 包含了应用描述和一系列用于描述 Kubernetes 资源的模板文件。Chart 可以被打包到一个 Chart 仓库中，供其他用户下载和使用。通过 Chart，用户可以方便地创建、安装、升级和管理 Kubernetes 应用。

总之，Helm v3 通过简化 Helm 命令的执行过程和摆脱对 Tiller 服务器的依赖，进一步提高了 Helm 的易用性和可扩展性。同时，Helm v3 仍然保持了对旧版本 Helm 命令的支持，使得用户可以平滑地迁移到新版本。

2、安装 Helm v3

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

K8s 版本支持的各个 helm 版本对照表：

官方网址：https://helm.sh/zh/docs/topics/version_skew/

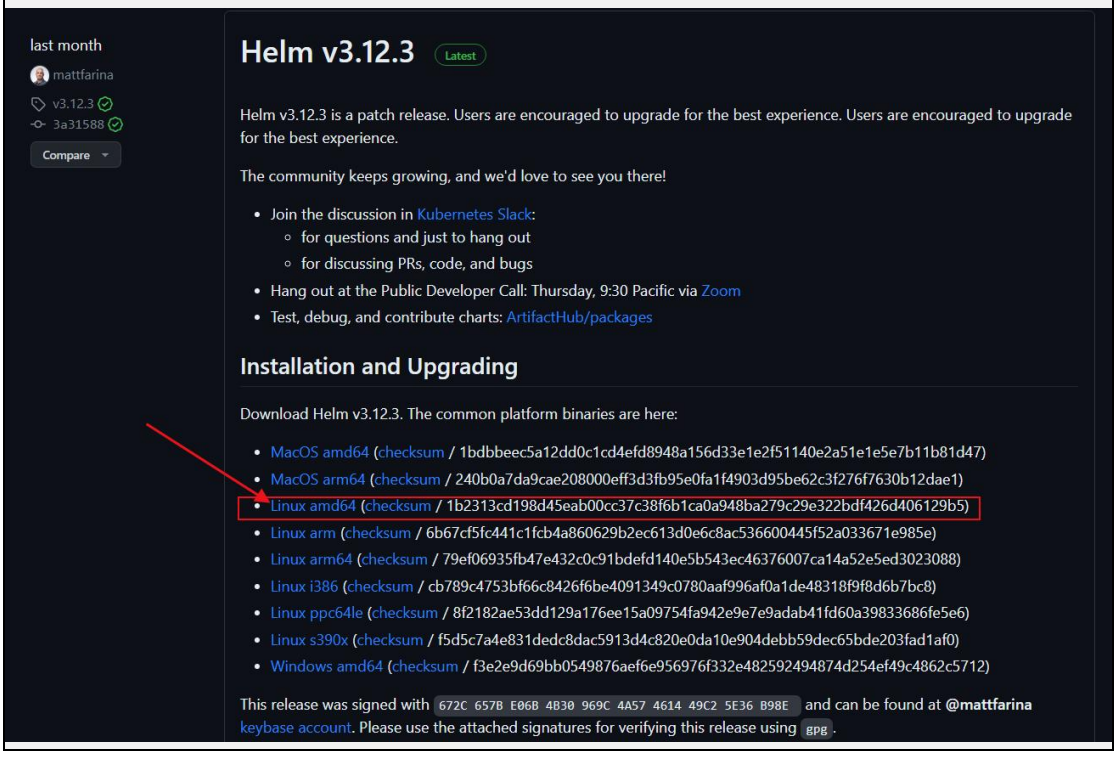
Helm 版本	支持的 Kubernetes 版本
3.12.x	1.27.x - 1.24.x
3.11.x	1.26.x - 1.23.x
3.10.x	1.25.x - 1.22.x
3.9.x	1.24.x - 1.21.x

我们当前是 k8s 1.26.x 版本，那么我们应该选择 3.12.x 或 3.11.x 的 helm 版本。

下载 Helm v3：

下载地址：https://github.com/helm/helm/releases

下载：



上传软件包到 k8s 管理节点、解压、安装：

```
1、上传软件包到 k8s 管理节点
    [root@k8s-master01 ~]# ll helm-v3.12.3-linux-amd64.tar.gz
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
    -rw-r--r-- 1 root root 16028423 Sep  4 04:41 helm-v3.12.3-linux-amd64.tar.gz

2、解压
    [root@k8s-master01 ~]# tar xf helm-v3.12.3-linux-amd64.tar.gz

3、安装
    [root@k8s-master01 ~]# mv linux-amd64/helm /usr/local/bin/
    mv: overwrite ‘/usr/local/bin/helm’ ? y

4、查看 helm 版本
    [root@k8s-master01 ~]# helm version
    version.BuildInfo{Version:"v3.12.3",
GitCommit:"3a31588ad33fe3b89af5a2a54ee1d25bfe6eaa5e", GitTreeState:"clean",
GoVersion:"go1.20.7"}
```

3、Helm 常用命令

3.1、Helm 自动补全

Helm completion 是 Helm 客户端的一个功能，用于自动补全 Helm 命令和参数。它可以帮助用户更方便地使用 Helm 命令，提高工作效率。

要使用 Helm completion 功能，需要先确保已经安装了 Helm 客户端，并根据您的操作系统和 Kubernetes 环境进行了正确配置。

当配置了 Helm 自动补全后，在执行 Helm 命令时，使用 Tab 键可以触发自动补全功能。

- 语法说明

```
    [root@k8s-master01 ~]# helm completion -h
    Usage:
      helm completion [command]

    Available Commands:
      bash：为 bash 生成自动补全脚本
      fish：为 fish 生成自动补全脚本
      powershell：为 powershell 生成自动补全脚本
      zsh：为 zsh 生成自动补全脚本
```

- 为 linux 系统（bash）配置自动补全

```
1、安装 helm 补全
    [root@k8s-master01 ~]# helm completion bash > /etc/bash_completion.d/helm
    [root@k8s-master01 ~]# source /etc/bash_completion.d/helm
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

2、测试补全

```
# 测试补全，发现报错
[root@k8s-master01 ~]# helm s`tab`
-bash: _get_comp_words_by_ref: command not found
```

3、这是因为 helm 和 kubectl 命令自动补全需要安装 bash-completion

```
[root@k8s-master01 ~]# yum -y install bash-completion
```

4、再次测试补全

```
[root@k8s-master01 ~]# helm s
```

```
[root@k8s-master01 ~]# helm s
search   (search for a keyword in charts)          status  (display the status of the named release)
show     (show information of a chart)
[root@k8s-master01 ~]# helm s
```

3.2、Helm 仓库管理命令

常用仓库地址：

- 阿里云仓库：
  https://kubernetes.oss-cn-hangzhou.aliyuncs.com/charts

- bitnami 仓库：
  https://charts.bitnami.com/bitnami

- 官方仓库：
  https://hub.kubeapps.com/charts/incubator
  官方 chart 仓库，国内可能无法访问。

- 微软仓库：
  http://mirror.azure.cn/kubernetes/charts/
  这个仓库推荐，基本上官网有的 chart 这里都有，国内可能无法访问。

- 应用示例

示例 1：添加仓库

```
1、添加阿里云的 chart 仓库
    [root@k8s-master01 ~]# helm repo add aliyun
https://kubernetes.oss-cn-hangzhou.aliyuncs.com/charts
    "aliyun" has been added to your repositories

2、添加 bitnami 的 chart 仓库
    [root@k8s-master01 ~]# helm repo add bitnami
https://charts.bitnami.com/bitnami
    "bitnami" has been added to your repositories
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

示例 2：更新 chart 仓库

```
[root@k8s-master01 ~]# helm repo update
```

为什么要更新 chart 仓库？

执行 helm repo update 命令的目的是为了更新 Helm 客户端的本地仓库索引文件。当你执行这个命令时，Helm 客户端会从 Helm 仓库中获取最新的 chart 信息，并更新本地的索引文件。这样，你就可以使用 helm search 命令查看到最新的 chart 信息，也可以使用 helm install 命令安装最新的 chart 版本。

更新本地仓库索引文件的好处在于，你可以在不必知道具体的 chart 版本号的情况下，使用 helm search 命令查找最新的、适合你需求的 chart。另外，当你执行 helm install 命令时，Helm 客户端会根据本地索引文件中的信息，自动选择合适的 chart 版本进行安装，避免了手动指定 chart 版本号的问题。

因此，建议定期执行 helm repo update 命令来保持本地仓库索引文件的最新状态，以便更好地管理和部署 Kubernetes 应用。

示例 3：查看仓库

```
[root@k8s-master01 ~]# helm repo list
NAME    URL
aliyun  https://kubernetes.oss-cn-hangzhou.aliyuncs.com/charts
bitnami https://charts.bitnami.com/bitnami
```

示例 4：删除 chart 仓库

```
[root@k8s-master01 ~]# helm repo remove aliyun
```

这个步骤，大家知道即可，这里暂不操作。你若执行删除操作，重新再添加阿里云 chart 仓库地址即可。

示例 5：查看指定仓库中的 chart

1、查看阿里云仓库中的 chart

```
[root@k8s-master01 ~]# helm search repo aliyun
```

```
[root@k8s-master01 ~]# helm search repo aliyun
NAME                            CHART VERSION   APP VERSION     DESCRIPTION
aliyun/acs-engine-autoscaler    2.1.3           2.1.1           Scales worker nodes within agent pools
aliyun/aerospike                0.1.7           v3.14.1.2       A Helm chart for Aerospike in Kubernetes
aliyun/anchore-engine           0.1.3           0.1.6           Anchore container analysis and policy evaluatio...
aliyun/artifactory              7.0.3           5.8.4           Universal Repository Manager supporting all maj...
aliyun/artifactory-ha           0.1.0           5.8.4           Universal Repository Manager supporting all maj...
aliyun/aws-cluster-autoscaler   0.3.2                           Scales worker nodes within autoscaling groups.
aliyun/bitcoind                 0.1.0           0.15.1          Bitcoin is an innovative payment network and a ...
aliyun/buildkite                0.2.1           3               Agent for Buildkite
aliyun/centrifugo               2.0.0           1.7.3           Centrifugo is a real-time messaging server.
aliyun/cert-manager             0.2.2           0.2.3           A Helm chart for cert-manager
aliyun/chaoskube                0.6.2           0.6.1           Chaoskube periodically kills random pods in you...
aliyun/chronograf               0.4.2                           Open-source web application written in Go and R...
aliyun/cluster-autoscaler       0.4.2           1.1.0           Scales worker nodes within autoscaling groups.
aliyun/cockroachdb              0.6.5           1.1.5           CockroachDB is a scalable, survivable, strongly...
```

3.3、搜索 Chart

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

搜索会读取系统上配置的所有仓库，并查找匹配。搜索这些仓库会使用存储在系统中的元数据。

- 应用示例

示例 1：搜索与关键字“nginx”匹配的最新版本

```
[root@k8s-master01 ~]# helm search repo nginx
```

```
[root@k8s-master01 ~]# helm search repo nginx
NAME                                    CHART VERSION   APP VERSION     DESCRIPTION
aliyun/nginx-ingress                    0.9.5           0.10.2          An nginx Ingress controller that uses ConfigMap...
aliyun/nginx-lego                       0.3.1                           Chart for nginx-ingress-controller and kube-lego
bitnami/nginx                           15.2.0          1.25.2          NGINX Open Source is a web server that can be a...
bitnami/nginx-ingress-controller        9.8.0           1.8.1           NGINX Ingress Controller is an Ingress controll...
bitnami/nginx-intel                     2.1.15          0.4.9           DEPRECATED NGINX Open Source for Intel is a lig...
aliyun/gcloud-endpoints                 0.1.0                           Develop, deploy, protect and monitor your APIs ...
[root@k8s-master01 ~]#
```

示例 2：搜索指定 chart 所有版本

```
[root@k8s-master01 ~]# helm search repo bitnami/nginx --versions
```

```
[root@k8s-master01 ~]# helm search repo bitnami/nginx --versions
NAME                    CHART VERSION   APP VERSION     DESCRIPTION
bitnami/nginx           15.2.0          1.25.2          NGINX Open Source is a web server that can be a...
bitnami/nginx           15.1.5          1.25.2          NGINX Open Source is a web server that can be a...
bitnami/nginx           15.1.4          1.25.2          NGINX Open Source is a web server that can be a...
bitnami/nginx           15.1.3          1.25.2          NGINX Open Source is a web server that can be a...
bitnami/nginx           15.1.2          1.25.1          NGINX Open Source is a web server that can be a...
bitnami/nginx           15.1.1          1.25.1          NGINX Open Source is a web server that can be a...
bitnami/nginx           15.1.0          1.25.1          NGINX Open Source is a web server that can be a...
bitnami/nginx           15.0.2          1.25.1          NGINX Open Source is a web server that can be a...
bitnami/nginx           15.0.1          1.25.0          NGINX Open Source is a web server that can be a...
bitnami/nginx           15.0.0          1.25.0          NGINX Open Source is a web server that can be a...
bitnami/nginx           14.2.2          1.24.0          NGINX Open Source is a web server that can be a...
bitnami/nginx           14.2.1          1.24.0          NGINX Open Source is a web server that can be a...
bitnami/nginx           14.1.1          1.24.0          NGINX Open Source is a web server that can be a...
bitnami/nginx           14.1.0          1.24.0          NGINX Open Source is a web server that can be a...
bitnami/nginx           14.0.0          1.24.0          NGINX Open Source is a web server that can be a...
bitnami/nginx           13.2.34         1.23.4          NGINX Open Source is a web server that can be a...
bitnami/nginx           13.2.33         1.23.4          NGINX Open Source is a web server that can be a...
bitnami/nginx           13.2.32         1.23.4          NGINX Open Source is a web server that can be a...
bitnami/nginx           13.2.31         1.23.3          NGINX Open Source is a web server that can be a...
bitnami/nginx           13.2.30         1.23.3          NGINX Open Source is a web server that can be a...
bitnami/nginx           13.2.29         1.23.3          NGINX Open Source is a web server that can be a...
bitnami/nginx           13.2.28         1.23.3          NGINX Open Source is a web server that can be a...
```

示例 3：查看 chart 详细信息

```
[root@k8s-master01 ~]# helm show chart bitnami/nginx
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
[root@k8s-master01 ~]# helm show chart bitnami/nginx
annotations:
  category: Infrastructure
  images: |
    - name: git
      image: docker.io/bitnami/git:2.41.0-debian-11-r76
    - name: nginx-exporter
      image: docker.io/bitnami/nginx-exporter:0.11.0-debian-11-r324
    - name: nginx
      image: docker.io/bitnami/nginx:1.25.2-debian-11-r3
  licenses: Apache-2.0
apiVersion: v2
appVersion: 1.25.2
dependencies:
- name: common
  repository: oci://registry-1.docker.io/bitnamicharts
  tags:
  - bitnami-common
  version: 2.x.x
description: NGINX Open Source is a web server that can be also used as a reverse
  proxy, load balancer, and HTTP cache. Recommended for high-demanding sites due to
  its ability to provide faster content.
home: https://bitnami.com
icon: https://bitnami.com/assets/stacks/nginx/img/nginx-stack-220x234.png
keywords:
- nginx
- http
- web
- www
- reverse proxy
maintainers:
- name: VMware, Inc.
  url: https://github.com/bitnami/charts
name: nginx
sources:
- https://github.com/bitnami/charts/tree/main/bitnami/nginx
version: 15.2.0
```

从上图我们可以看到这个 chart 的版本、所使用的镜像、github 地址等等信息。

3.4、拉取 Chart

helm pull 可以拉取指定 chart 到本地。

- 语法说明

```
[root@k8s-master01 ~]# helm pull
Usage:
  helm pull [chart URL | repo/chartname] [...] [flags]

Aliases:
  pull, fetch：别名，pull 等同于 fetch

Flags:
  --version string：拉取指定版本，如果不指定则拉取最新版本。
```

- 应用示例

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

示例 1：下载最新的 chart

```
[root@k8s-master01 ~]# helm pull bitnami/nginx
[root@k8s-master01 ~]# ll nginx-15.2.0.tgz
-rw-r--r-- 1 root root 38645 Sep  4 05:54 nginx-15.2.0.tgz
```

示例 2：下载指定版本的 chart

```
[root@k8s-master01 ~]# helm pull bitnami/nginx --version 15.1.0
[root@k8s-master01 ~]# ll nginx-15.1.0.tgz
-rw-r--r-- 1 root root 38139 Sep  4 05:55 nginx-15.1.0.tgz
```

3.5、安装 Chart

要通过 chart 包部署创建 kubernetes 资源，可以使用 helm install 命令进行资源的创建。

安装 chart 的方式比较多，如下：

```
1、通过 chart 包：helm install mynginx ./nginx-1.2.3.tgz
2、通过未打包 chart 目录的路径：helm install mynginx .
3、通过 URL 绝对路径：helm install mynginx
https://example.com/charts/nginx-1.2.3.tgz
```

- 应用案例

示例 1：解压安装 nginx-15.1.0.tgz chart

```
1、创建目录、下载 chart、解压
    [root@k8s-master01 ~]# mkdir /test
    [root@k8s-master01 ~]# cd /test/
    [root@k8s-master01 test]# helm pull bitnami/nginx --version 15.1.0
    [root@k8s-master01 test]# tar xf nginx-15.1.0.tgz

2、查看解压后的目录结构
    [root@k8s-master01 test]# cd nginx/
    [root@k8s-master01 nginx]# ll
    total 104
    -rw-r--r-- 1 root root   225 Jun 29 04:25 Chart.lock
    drwxr-xr-x 3 root root    20 Sep  4 06:00 charts
    -rw-r--r-- 1 root root   757 Jun 29 04:25 Chart.yaml
    -rw-r--r-- 1 root root 49208 Jun 29 04:25 README.md
    drwxr-xr-x 2 root root   325 Sep  4 06:00 templates
    -rw-r--r-- 1 root root  2225 Jun 29 04:25 values.schema.json
    -rw-r--r-- 1 root root 36989 Jun 29 04:25 values.yaml

对上面的目录结构说明如下：
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

Chart.lock：这个文件通常用于锁定 Chart 的依赖关系，以防止在更新或安装期间发生冲突。

charts：这是一个目录，可能包含其他 Chart 文件或子目录。

Chart.yaml：这是 Chart 的主配置文件，其中包含有关 Chart 的元数据，例如名称、描述、版本和依赖项。

README.md：这是一个 Markdown 格式的文件，通常用于提供关于目录或项目的文档。

templates：这是一个目录，可能包含用于 Kubernetes 配置的模板文件。

values.schema.json：这是一个 JSON 格式的文件，包含 Chart 的值的验证和默认值。

values.yaml:这是一个 YAML 格式的文件,包含在安装 Chart 时用于定义配置的默认值。

3、安装 chart

[root@k8s-master01 nginx]# helm install nginx .

```
[root@k8s-master01 nginx]# helm install nginx .
NAME: nginx
LAST DEPLOYED: Mon Sep  4 06:14:27 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: nginx
CHART VERSION: 15.1.0
APP VERSION: 1.25.1

** Please be patient while the chart is being deployed **
NGINX can be accessed through the following DNS name from within your cluster:

    nginx.default.svc.cluster.local (port 80)

To access NGINX from outside the cluster, follow the steps below:

1. Get the NGINX 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 nginx'

    export SERVICE_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0].port}" services nginx)
    export SERVICE_IP=$(kubectl get svc --namespace default nginx -o jsonpath='{.status.loadBalancer.ingress[0].ip}')
    echo "http://${SERVICE_IP}:${SERVICE_PORT}"
[root@k8s-master01 nginx]#
```

4、查看 pod 和 svc

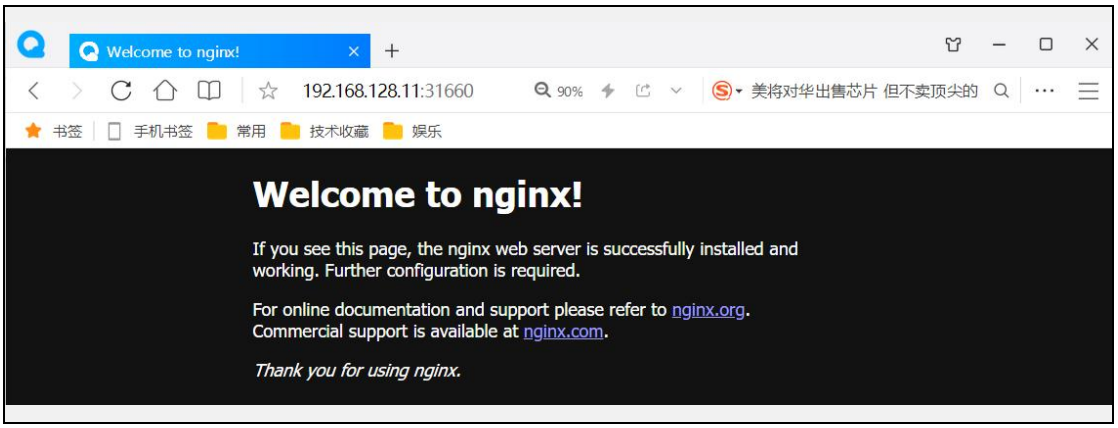
```
[root@k8s-master01 nginx]# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
nginx-77b4f44555-h2f4c   1/1     Running   0          34s

[root@k8s-master01 nginx]# kubectl get svc nginx
NAME    TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
nginx   LoadBalancer   10.10.228.47   <pending>     80:31660/TCP   40s
```

5、浏览器访问测试

浏览器输入：“http://192.168.128.11:31660”，访问结果如下图所示：

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。



3.6、查看安装的 Chart

helm list 命令用于列出所有已经发布到 Kubernetes 集群的 release。这个命令会显示 release 的名称以及相应的版本号。

helm list 命令的常用参数包括：

```
-a, --all：显示所有 release，包括正在升级的 release。
-A, --all-namespaces：显示所有命名空间的 release
-m, --max：限制返回的 release 数量。可以设置为一个整数，例如--max 10。
--offset：与--all 参数一起使用，表示从哪个 release 开始显示。可以用于分页显示结果。
--uninstalled：显示已经释放但未安装的 release。
--installed：显示已经安装的 release。
--failed：显示安装失败的 release。
```

● 应用示例

示例 1：查看所有命名空间的 release

```
[root@k8s-master01 ~]# helm list -A
```

```
[root@k8s-master01 ~]# helm list -A
NAME    NAMESPACE       REVISION        UPDATED                                 STATUS          CHART                   APP VERSION
calico  kube-system     1               2023-05-24 06:02:30.080216558 -0400 EDT deployed        tigera-operator-v3.25.1 v3.25.1
nginx   default         1               2023-09-04 06:14:27.04124339 -0400 EDT  deployed        nginx-15.1.0            1.25.1
[root@k8s-master01 ~]#
```

3.7、卸载 Chart

要卸载 Helm Chart，可以使用 helm uninstall 命令。

● 应用示例

示例 1：卸载上面安装的 nginx

```
1、查看所有 release
    [root@k8s-master01 ~]# helm list -A
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
[root@k8s-master01 ~]# helm list -A
NAME    NAMESPACE       REVISION        UPDATED                                 STATUS          CHART                   APP VERSION
calico  kube-system     1               2023-05-24 06:02:30.080216558 -0400 EDT deployed        tigera-operator-v3.25.1 v3.25.1
nginx   default         1               2023-09-04 06:14:27.04124339 -0400 EDT  deployed        nginx-15.1.0            1.25.1
[root@k8s-master01 ~]#
```

2、卸载 nginx release

```
[root@k8s-master01 ~]# helm uninstall nginx
release "nginx" uninstalled
```

3、再次查看所有 release

```
[root@k8s-master01 ~]# helm list -A
```

```
[root@k8s-master01 ~]# helm list -A
NAME    NAMESPACE       REVISION        UPDATED                                 STATUS          CHART                   APP VERSION
calico  kube-system     1               2023-05-24 06:02:30.080216558 -0400 EDT deployed        tigera-operator-v3.25.1 v3.25.1
[root@k8s-master01 ~]#
```

4、Chart 模板指南

4.1、第一个 Chart

（1）生成 Chart 模板

当我们安装好 helm 之后我们可以开始自定义 chart，那么我们需要先创建出一个模板如下：

1、生成 chart 模板，文件夹名称叫 mychart

```
[root@k8s-master01 ~]# helm create mychart
Creating mychart
```

2、查看生成的目录结构

```
[root@k8s-master01 ~]# cd mychart/
[root@k8s-master01 mychart]# tree .
.
├── charts  #用于存放所依赖的子 chart
├── Chart.yaml  #描述这个 Chart 的相关信息、包括名字、描述信息、版本等
├── templates  #模板目录，保留创建 k8s 的资源清单文件
│   ├── deployment.yaml  #deployment 资源的 go 模板文件
│   ├── _helpers.tpl  #模板助手文件，定义的值可在模板中使用
│   ├── hpa.yaml  #水平 pod 自动扩缩容 go 模板文件
│   ├── ingress.yaml  #七层代理 go 模板文件
│   ├── NOTES.txt  #存放提示信息的文件
│   ├── serviceaccount.yaml  #sa 模板文件
│   ├── service.yaml  #service 的 go 模板文件
│   └── tests  #用于测试的文件
│       └── test-connection.yaml
└── values.yaml  #模板的值文件，这些值会在安装时应用到 GO 模板生成部署文件
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

（2）删除生成的 template

```
[root@k8s-master01 mychart]# rm -rf templates/*
```

编制生产环境级别的 chart 时，有这些 chart 的基础版本会很有用。因此在日常编写中，可能不需要删除它们。

（3）第一个模板

这里写一个 configmap 模板，就是我们平时所写的资源清单文件。

```
[root@k8s-master01 mychart]# vi templates/configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mychart-configmap
data:
  myvalue: "Hello World"
```

（4）安装 Chart

```
1、安装 chart
[root@k8s-master01 mychart]# helm install test-configmap .
NAME: test-configmap
LAST DEPLOYED: Mon Sep  4 21:41:17 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None

2、使用 helm 检索版本并查看实际加载的模板
[root@k8s-master01 mychart]# helm get manifest test-configmap
---
# Source: mychart/templates/configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mychart-configmap
data:
  myvalue: "Hello World"
```

（5）卸载 Chart

```
1、查看 chart
[root@k8s-master01 mychart]# helm list
[root@k8s-master01 mychart]# helm list
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP VERSION
test-configmap  default         1               2023-09-04 21:41:17.498576882 -0400 EDT deployed        mychart-0.1.0   1.16.0
[root@k8s-master01 mychart]#
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
2、卸载 chart
    [root@k8s-master01 mychart]# helm uninstall test-configmap
    release "test-configmap" uninstalled
```

4.2、内置对象

上面我们写的 configmap 模板比较固定，不够灵活，假如我们想要在安装时，安装到 kube-system 命名空间下，configmap name 我们也想自定义。那么我们不能每次在安装时修改模板文件吧，那就太麻烦了。

（1）修改模板

```
1、这里我们引入两个常用的内置对象
    [root@k8s-master01 mychart]# vi templates/configmap.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: {{ .Release.Name }}-configmap
      namespace: {{ .Release.Namespace }}
    data:
      myvalue: "Hello World"
```

对上面的内置变量说明如下：

```
.Release.Name 和.Release.Namespace 都是 Helm 的内置变量。分别说明如下：
    Release.Name：release 名称
    Release.Namespace：版本中包含的命名空间
```

（2）安装部署、检查、卸载

```
1、安装 chart
    [root@k8s-master01 mychart]# helm install test . -n kube-system
    NAME: test
    LAST DEPLOYED: Mon Sep  4 22:17:59 2023
    NAMESPACE: kube-system
    STATUS: deployed
    REVISION: 1
    TEST SUITE: None

    说明：因为我们这里的 NAMESPACE 为 kube-system，会将这个值传递给
Release.Namespace，所以这也是为什么 configmap 会部署在 kube-system 命名空间下。

2、使用 helm 检索版本并查看实际加载的模板
    [root@k8s-master01 mychart]# helm get manifest test -n kube-system
    ---
    # Source: mychart/templates/configmap.yaml
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: test-configmap
      namespace: kube-system
    data:
      myvalue: "Hello World"

3、卸载 chart
    [root@k8s-master01 mychart]# helm uninstall test -n kube-system
    release "test" uninstalled
```

4.3、Value 文件

在上一部分我们了解了 Helm 模板提供的内置对象。对于很多参数，我们不能只通过命令行传入，我们可以定义模板使用 value 文件中的值。

在 values.yaml 中的值，我们在 template 模板中可以使用“{{ .Values.值 }}”来进行调用。

（1）清空默认的 value 文件内容

```
    [root@k8s-master01 mychart]# > values.yaml
```

（2）修改 value 文件内容

```
    [root@k8s-master01 mychart]# vi values.yaml
    Namespace: kube-system
    Index: This is test web
```

（3）修改 configmap template 文件内容

```
    [root@k8s-master01 mychart]# vi templates/configmap.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: {{ .Release.Name }}-configmap
      namespace: {{ .Values.Namespace }}
    data:
      myvalue: "Hello World"
      index: "{{ .Values.Index }}"
```

这里我们希望 index 对应的值能被双引号引起来，这里使用的方式比较传统。后面我们可以用 quote 函数自动添加双引号。

（4）安装部署、检查、卸载

```
1、安装 chart
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
    [root@k8s-master01 mychart]# helm install test .

2、使用 helm 检索版本并查看实际加载的模板
    [root@k8s-master01 mychart]# helm get manifest test
    ---
    # Source: mychart/templates/configmap.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: test-configmap
      namespace: kube-system
    data:
      myvalue: "Hello World"
      index: "This is test web"

3、卸载 chart
    [root@k8s-master01 mychart]# helm uninstall test
    release "test" uninstalled
```

4.4、模板函数

到目前为止，我们已经知道了如何将信息传到模板中。但是传入的信息并不能被修改。有时我们希望以一种更有用的方式来转换所提供的数据。

官方模块函数手册地址：
https://helm.sh/zh/docs/chart_template_guide/function_list/

4.4.1、quote 函数

通过调用模板指令中的 quote 函数把.Values 对象中的字符串属性用引号引起来，然后放到模板中。

- 应用示例

```
1、修改 configmap template 文件
    [root@k8s-master01 mychart]# vi templates/configmap.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: {{ .Release.Name }}-configmap
      namespace: {{ .Values.Namespace }}
    data:
      myvalue: "Hello World"
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
    index: {{ quote .Values.Index }}

2、安装 chart
    [root@k8s-master01 mychart]# helm install test .

3、使用 helm 检索版本并查看实际加载的模板
    [root@k8s-master01 mychart]# helm get manifest test
    ---
    # Source: mychart/templates/configmap.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: test-configmap
      namespace: kube-system
    data:
      myvalue: "Hello World"
      index: "This is test web"

4、卸载 chart
    [root@k8s-master01 mychart]# helm uninstall test
    release "test" uninstalled
```

4.4.2、default 函数

模板中频繁使用的一个函数是 default。这个函数允许你在模板中指定一个默认值，以防这个值被忽略。

● 应用示例

```
1、修改 configmap template 文件
    [root@k8s-master01 mychart]# vi templates/configmap.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: {{ .Release.Name }}-configmap
      namespace: {{ .Values.Namespace }}
    data:
      myvalue: "Hello World"
      index: {{ .Values.Indexx | default "Welcome to visit" | quote }}

    说明：如果.Values.Indexx 不存在时，那么此时默认值为“Welcome to visit”。如果.Values.Indexx 存在，则 default 不生效，值为“.Values.Indexx”所对应的值。
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
2、安装 chart
    [root@k8s-master01 mychart]# helm install test .

3、使用 helm 检索版本并查看实际加载的模板
    [root@k8s-master01 mychart]# helm get manifest test
    ---
    # Source: mychart/templates/configmap.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: test-configmap
      namespace: kube-system
    data:
      myvalue: "Hello World"
      index: "Welcome to visit"

4、卸载 chart
    [root@k8s-master01 mychart]# helm uninstall test
    release "test" uninstalled
```

4.5、流控制

Helm 的流控制主要包括以下几种控制结构：

if/else：创建条件语句。

with：指定范围。

range：提供“for each”类型的循环。

这些控制结构可以让你更灵活地控制 Helm 模板的迭代流。

4.5.1、IF/Else 条件语句

if/else 语法如下：

```
{{- if PIPELINE }}
  # Do something
{{- else if OTHER PIPELINE }}
  # Do something else
{{- else }}
  # Default case
{{- end }}
```

- 应用示例

```
1、修改 configmap template 文件
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
    # 判断“.Values.Namespace”的值是否等于 kube-system，如果等于就为 true 否则为 false。
    [root@k8s-master01 mychart]# vi templates/configmap.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: {{ .Release.Name }}-configmap
      namespace: {{ .Values.Namespace }}
    data:
      myvalue: "Hello World"
      index: {{ .Values.Indexx | default "Welcome to visit" | quote }}
      {{- if eq .Values.Namespace "kube-system" }}
      mug: "true"
      {{- else }}
      mug: "false"
      {{- end -}}

2、安装 chart
    [root@k8s-master01 mychart]# helm install test .

3、使用 helm 检索版本并查看实际加载的模板
    [root@k8s-master01 mychart]# helm get manifest test
    ---
    # Source: mychart/templates/configmap.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: test-configmap
      namespace: kube-system
    data:
      myvalue: "Hello World"
      index: "Welcome to visit"
      mug: "true"

4、卸载 chart
    [root@k8s-master01 mychart]# helm uninstall test
    release "test" uninstalled
```

4.5.2、with 指定范围

这个用来控制变量范围。回想一下，.是对 当前作用域 的引用。因此 .Values 就是告诉模板在当前作用域查找 Values 对象。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

● 应用示例

1、修改 values.yaml 文件

```
[root@k8s-master01 mychart]# vi values.yaml
Namespace: kube-system
Index: This is test web
pod:
  images: nginx
  tag: latest
```

2、修改 configmap template 文件

```
[root@k8s-master01 mychart]# vi templates/configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
  namespace: {{ .Values.Namespace }}
data:
  myvalue: "Hello World"
  index: {{ .Values.Indexx | default "Welcome to visit" | quote }}
  {{- with .Values.pod }}
  image: "{{.images}}:{{.tag | default "latest"}}"
  {{- end }}
```

2、安装 chart

```
[root@k8s-master01 mychart]# helm install test .
```

3、使用 helm 检索版本并查看实际加载的模板

```
[root@k8s-master01 mychart]# helm get manifest test
---
# Source: mychart/templates/configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap
  namespace: kube-system
data:
  myvalue: "Hello World"
  index: "Welcome to visit"
  image: "nginx:latest"
```

4、卸载 chart

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
[root@k8s-master01 mychart]# helm uninstall test
release "test" uninstalled
```

4.5.3、Range 循环

很多编程语言支持使用 for 循环，foreach 循环，或者类似的方法机制。 在 Helm 的模板语言中，在一个集合中迭代的方式是使用 range 操作符。

● 应用示例

1、修改 values.yaml 文件

```
[root@k8s-master01 mychart]# vi values.yaml
Namespace: kube-system
Index: This is test web
pod:
  images: nginx
  tag: latest
color:
  - red
  - green
  - blue
```

2、修改 configmap template 文件

```
[root@k8s-master01 mychart]# vi templates/configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
  namespace: {{ .Values.Namespace }}
data:
  myvalue: "Hello World"
  index: {{ .Values.Indexx | default "Welcome to visit" | quote }}
  color: |-
    {{- range $.Values.color }}
    - {{ . | title | quote }}
    {{- end }}
```

2、安装 chart

```
[root@k8s-master01 mychart]# helm install test .
```

3、使用 helm 检索版本并查看实际加载的模板

```
[root@k8s-master01 mychart]# helm get manifest test
---
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
    # Source: mychart/templates/configmap.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: test-configmap
      namespace: kube-system
    data:
      myvalue: "Hello World"
      index: "Welcome to visit"
      colour: |-
        - "Red"
        - "Green"
        - "Blue"

4、卸载 chart
    [root@k8s-master01 mychart]# helm uninstall test
    release "test" uninstalled
```

4.6、变量

变量在模板中，很少被使用。但是我们可以使用变量简化代码，并更好地使用 with 和 range。

● 应用示例

示例 1：直接赋值

```
1、修改 configmap template 文件
    [root@k8s-master01 mychart]# vi templates/configmap.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: {{ .Release.Name }}-configmap
      namespace: {{ .Values.Namespace }}
    data:
      myvalue: "Hello World"
      index: {{ .Values.Indexx | default "Welcome to visit" | quote }}
      {{- $image_tag := .Values.pod.tag }}
      image_tag: {{ $image_tag }}

2、安装 chart
    [root@k8s-master01 mychart]# helm install test .

3、使用 helm 检索版本并查看实际加载的模板
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
    [root@k8s-master01 mychart]# helm get manifest test
    ---
    # Source: mychart/templates/configmap.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: test-configmap
      namespace: kube-system
    data:
      myvalue: "Hello World"
      index: "Welcome to visit"
      image_tag: latest

4、卸载 chart
    [root@k8s-master01 mychart]# helm uninstall test
    release "test" uninstalled
```

4.7、命名模板

● 局部的和_文件

目前为止，我们已经使用了单个文件，且单个文件中包含了单个模板。但 Helm 的模板语言允许你创建命名的嵌入式模板，这样就可以在其他位置按名称访问。

在编写模板细节之前，文件的命名惯例需要注意：

```
    1、templates/中的大多数文件被视为包含 Kubernetes 清单
    2、NOTES.txt 是个例外
    3、命名以下划线(_)开始的文件则假定没有包含清单内容。这些文件不会渲染为 Kubernetes 对象定义，但在其他 chart 模板中都可用。
```

这些文件用来存储局部和辅助对象，实际上当我们第一次创建 mychart 时，会看到一个名为_helpers.tpl 的文件，这个文件是模板局部的默认位置。

● 语法

define 操作允许我们在模板文件中创建一个命名模板，语法如下：

```
    {{- define "MY.NAME" }}
      # body of template here
    {{- end }}
```

● 应用示例

示例 1：用 define 和 template 声明和使用模板

```
1、定义一个模板封装 kubernetes 的标签
    [root@k8s-master01 mychart]# vi templates/_helpers.tpl
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
    {{- define "mychart.labels" }}
      labels:
        app: test
    {{- end }}

2、修改 configmap template 文件
    [root@k8s-master01 mychart]# vi templates/configmap.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: {{ .Release.Name }}-configmap
      namespace: {{ .Values.Namespace }}
      {{- template "mychart.labels" }}
    data:
      myvalue: "Hello World"
      index: {{ .Values.Indexx | default "Welcome to visit" | quote }}

3、安装 chart
    [root@k8s-master01 mychart]# helm install test .

3、使用 helm 检索版本并查看实际加载的模板
    [root@k8s-master01 mychart]# helm get manifest test
    ---
    # Source: mychart/templates/configmap.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: test-configmap
      namespace: kube-system
      labels:
        app: test
    data:
      myvalue: "Hello World"
      index: "Welcome to visit"

4、卸载 chart
    [root@k8s-master01 mychart]# helm uninstall test
    release "test" uninstalled
```

示例 2：用 define 和 include 声明和使用模板

```
1、添加一个模板
    [root@k8s-master01 mychart]# vi templates/_helpers.tpl
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
    {{- define "mychart.app" -}}
    app_index: {{ .Values.Index | quote }}
    {{- end -}}

2、修改 configmap template 文件
    [root@k8s-master01 mychart]# vi templates/configmap.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: {{ .Release.Name }}-configmap
      namespace: {{ .Values.Namespace }}
      {{- template "mychart.labels" }}
    data:
      myvalue: "Hello World"
      index: {{ .Values.Indexx | default "Welcome to visit" | quote }}
    {{ include "mychart.app" . | indent 2 }}

    提示：indent 表示缩进 2 个空格

3、安装 chart
    [root@k8s-master01 mychart]# helm install test .

3、使用 helm 检索版本并查看实际加载的模板
    [root@k8s-master01 mychart]# helm get manifest test
    ---
    # Source: mychart/templates/configmap.yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: test-configmap
      namespace: kube-system
      labels:
        app: test
    data:
      myvalue: "Hello World"
      index: "Welcome to visit"
      app_index: "This is test web"

4、卸载 chart
    [root@k8s-master01 mychart]# helm uninstall test
    release "test" uninstalled
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

相较于使用 template，在 helm 中使用 include 被认为是更好的方式，只是为了更好地处理 YAML 文档的输出格式。

4.8、NOTES.txt 文件

该部分会介绍为 chart 用户提供说明的 Helm 工具。在 helm install 或 helm upgrade 命令的最后，Helm 会打印出对用户有用的信息。 使用模板可以高度自定义这部分信息。

要在 chart 添加安装说明，只需创建 templates/NOTES.txt 文件即可。该文件是纯文本，但会像模板一样处理， 所有正常的模板函数和对象都是可用的。

1、创建一个简单的 NOTES.txt 文件：

```
[root@k8s-master01 mychart]# vi templates/NOTES.txt
Thank you for installing {{ .Chart.Name }}.

Your release is named {{ .Release.Name }}.

To learn more about the release, try:

  $ helm status {{ .Release.Name }}
  $ helm get all {{ .Release.Name }}
```

2、执行“helm install test .”结果如下：

```
[root@k8s-master01 mychart]# helm install test .
NAME: test
LAST DEPLOYED: Tue Sep  5 04:05:50 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing mychart.

Your release is named test.

To learn more about the release, try:

  $ helm status test
  $ helm get all test
```

使用 NOTES.txt 这种方式是给用户提供关于如何使用新安装的 chart 细节信息的好方法。尽管并不是必需的，强烈建议创建一个 NOTES.txt 文件。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

5、自定义 Chart 模板实战

实战：自定义 Chart 模板部署部署 nginx 应用。使用 deployment 和 service 资源。

（1）生成 chart 模板

```
[root@k8s-master01 ~]# helm create nginx
Creating nginx

[root@k8s-master01 ~]# rm -rf nginx/templates/*
[root@k8s-master01 ~]# > nginx/values.yaml
```

（2）创建 deployment 模板

```
[root@k8s-master01 ~]# vi nginx/templates/deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name }}
  labels:
    {{- toYaml .Values.labels | nindent 4 }}
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      {{- toYaml .Values.labels | nindent 6 }}
  template:
    metadata:
      labels:
        {{- toYaml .Values.labels | nindent 8 }}
    spec:
      containers:
        - name: {{ .Release.Name }}
          image: {{.Values.image.repository}}:{{.Values.image.tag | default
"latest" }}
          imagePullPolicy: {{ .Values.image.pullPolicy }}
          ports:
            - name: http
              containerPort: {{ .Values.service.port }}
              protocol: TCP
          livenessProbe:
            httpGet:
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
                path: /
                port: http
            readinessProbe:
              httpGet:
                path: /
                port: http
            resources:
              {{- toYaml .Values.resources | nindent 12 }}
```

对上面 yaml 文件说明如下：

```
    apiVersion: apps/v1
    kind: Deployment
    metadata:
      name: {{ .Release.Name }}  #定义了这个部署对象的名称，使用了模板语法，这里使用了 .Release.Name 变量，表示根据发布名称来设置部署名称。
      labels:
        {{- toYaml .Values.labels | nindent 4 }}  #将.Values.labels 转换为 YAML 格式的字符串，并缩进 4 个空格。这是使用模板语法来动态生成配置内容。
    spec:
      replicas: {{ .Values.replicaCount }}  #定义了副本数量，即要创建的应用程序实例的数量。这里使用了 .Values.replicaCount 变量，表示根据值来设置副本数量。
      selector:
        matchLabels:
          {{- toYaml .Values.labels | nindent 6 }}
      template:
        metadata:
          labels:
            {{- toYaml .Values.labels | nindent 8 }}
        spec:
          containers:
            - name: {{ .Release.Name }}
              image: {{ .Values.image.repository }}:{{ .Values.image.tag | default "latest" }}  #指定了容器的镜像地址，这里使用了 .Values.image.repository 和 .Values.image.tag 变量，并设置了默认值为 "latest"。
              imagePullPolicy: {{ .Values.image.pullPolicy }}
              ports:
                - name: http
                  containerPort: {{ .Values.service.port }}
                  protocol: TCP
              livenessProbe:
                httpGet:
                  path: /
                  port: http
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
            readinessProbe:
              httpGet:
                path: /
                port: http
            resources:
              {{- toYaml .Values.resources | nindent 12 }}  #
将 .Values.resources 转换为 YAML 格式的字符串，并缩进 12 个空格。这是用于设置容器的资源限制。
```

（3）创建 service 模板

```
[root@k8s-master01 ~]# vi nginx/templates/service.yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}
  labels:
    {{- toYaml .Values.labels | nindent 4 }}
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.port }}
      targetPort: http
      protocol: TCP
      name: http
  selector:
    {{- toYaml .Values.labels | nindent 4 }}
```

（4）创建 NOTES.txt 文件

```
[root@k8s-master01 ~]# vi nginx/templates/NOTES.txt
Thank you for installing {{ .Chart.Name }}.

Your release is named {{ .Release.Name }}.

Get the application URL by running these commands:
  kubectl patch service {{ .Release.Name }} -p '{"spec":{"type":"NodePort"}}'
  kubectl get svc {{ .Release.Name }}

To learn more about the release, try:
  $ helm status {{ .Release.Name }}
  $ helm get all {{ .Release.Name }}
```

（5）创建 values.yaml 文件

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
[root@k8s-master01 ~]# vi nginx/values.yaml
replicaCount: 1

image:
  repository: nginx
  pullPolicy: IfNotPresent
  tag: latest

service:
  type: ClusterIP
  port: 80

resources:
  limits:
    cpu: 500m
    memory: 1024Mi
  requests:
    cpu: 100m
    memory: 128Mi

labels:
  app: nginx
  version: latest
```

（6）安装 chart

```
[root@k8s-master01 ~]# helm install nginx nginx/
NAME: nginx
LAST DEPLOYED: Tue Sep  5 03:17:59 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing nginx.

Your release is named nginx.

Get the application URL by running these commands:
  kubectl patch service nginx -p '{"spec":{"type":"NodePort"}}'
  kubectl get svc nginx

To learn more about the release, try:
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
$ helm status nginx
$ helm get all nginx
```

（7）查看创建的资源

```
[root@k8s-master01 ~]# kubectl get pods -l app=nginx
NAME                     READY   STATUS    RESTARTS   AGE
nginx-86c698c67f-mtsgx   1/1     Running   0          75s

[root@k8s-master01 ~]# kubectl get svc -l app=nginx
NAME    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
nginx   ClusterIP   10.10.255.184   <none>        80/TCP    81s
```

6、Helm 高级命令

6.1、检查 values 语法格式

我们在上面自定义 Chart 时，可能会存在语法格式问题。在创建之前，我们可以使用“helm lint”进行语法检查。

● 应用示例

示例 1：检查 values 语法格式：

```
[root@k8s-master01 ~]# helm lint nginx/
==> Linting nginx/
[INFO] Chart.yaml: icon is recommended

1 chart(s) linted, 0 chart(s) failed
```

通过上面可以看到语法正确

6.2、升级 release

Release 升级通常是更新 values.yaml 文件，比如更新镜像、更新资源限制等等。

● 应用示例

示例 1：更新上面部署的 nginx svc 类型为 nodeport

（1）查看当前 nginx 的 svc

```
[root@k8s-master01 ~]# kubectl get svc -l app=nginx
NAME    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
nginx   ClusterIP   10.10.153.185   <none>        80/TCP    9s
```

（2）升级：（可以更新 yaml 文件，然后重新部署 release，也可以起到更新作用）

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
[root@k8s-master01 ~]# helm upgrade --set service.type="NodePort" nginx nginx
Release "nginx" has been upgraded. Happy Helming!
NAME: nginx
LAST DEPLOYED: Tue Sep  5 04:27:24 2023
NAMESPACE: default
STATUS: deployed
REVISION: 2
TEST SUITE: None
NOTES:
Thank you for installing nginx.

Your release is named nginx.

Get the application URL by running these commands:
  kubectl patch service nginx -p '{"spec":{"type":"NodePort"}}'
  kubectl get svc nginx

To learn more about the release, try:
  $ helm status nginx
  $ helm get all nginx
```

（1）查看当前 nginx 的 svc

```
[root@k8s-master01 ~]# kubectl get svc -l app=nginx
NAME    TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
nginx   NodePort   10.10.153.185   <none>        80:31627/TCP   3m10
```

6.3、回滚 release

回滚之前可以使用“helm history”查看历史版本，然后使用“helm rollback”进行版本回退。操作如下：

（1）查看历史版本

```
[root@k8s-master01 ~]# helm history nginx
```

```
[root@k8s-master01 ~]# helm history nginx
REVISION        UPDATED                         STATUS          CHART           APP VERSION     DESCRIPTION
1               Tue Sep  5 04:28:10 2023        superseded      nginx-0.1.0     1.16.0          Install complete
2               Tue Sep  5 04:31:19 2023        deployed        nginx-0.1.0     1.16.0          Upgrade complete
[root@k8s-master01 ~]#
```

（2）回滚 nginx 版本到版本 1

```
[root@k8s-master01 ~]# helm rollback nginx 1
Rollback was a success! Happy Helming!
```

（3）查看当前 myapp 的 svc

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
[root@k8s-master01 ~]# kubectl get svc -l app=nginx
NAME    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
nginx   ClusterIP   10.10.153.185   <none>        80/TCP     5m20s
```

6.4、打包 Chart

Helm package 命令还可以使用一些选项来定制 Chart 的打包方式。例如，可以使用--version 选项指定 Chart 的版本号，使用--app-version 选项指定应用程序的版本号等。

● 应用示例

示例 1：将 nginx helm 打成一个 Chart 包

```
[root@k8s-master01 ~]# helm package /root/nginx/
Successfully packaged chart and saved it to: /root/nginx-0.1.0.tgz

[root@k8s-master01 ~]# ll /root/nginx-0.1.0.tgz
-rw-r--r-- 1 root root 1236 Sep  5 04:35 /root/nginx-0.1.0.tgz
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。