Published in Level Up Coding

Md Shamim Follow
Nov 8, 2022 · 5 min read · Listen

Save

Helm — Dependencies

A Deep Dive Into Helm Chart Dependencies

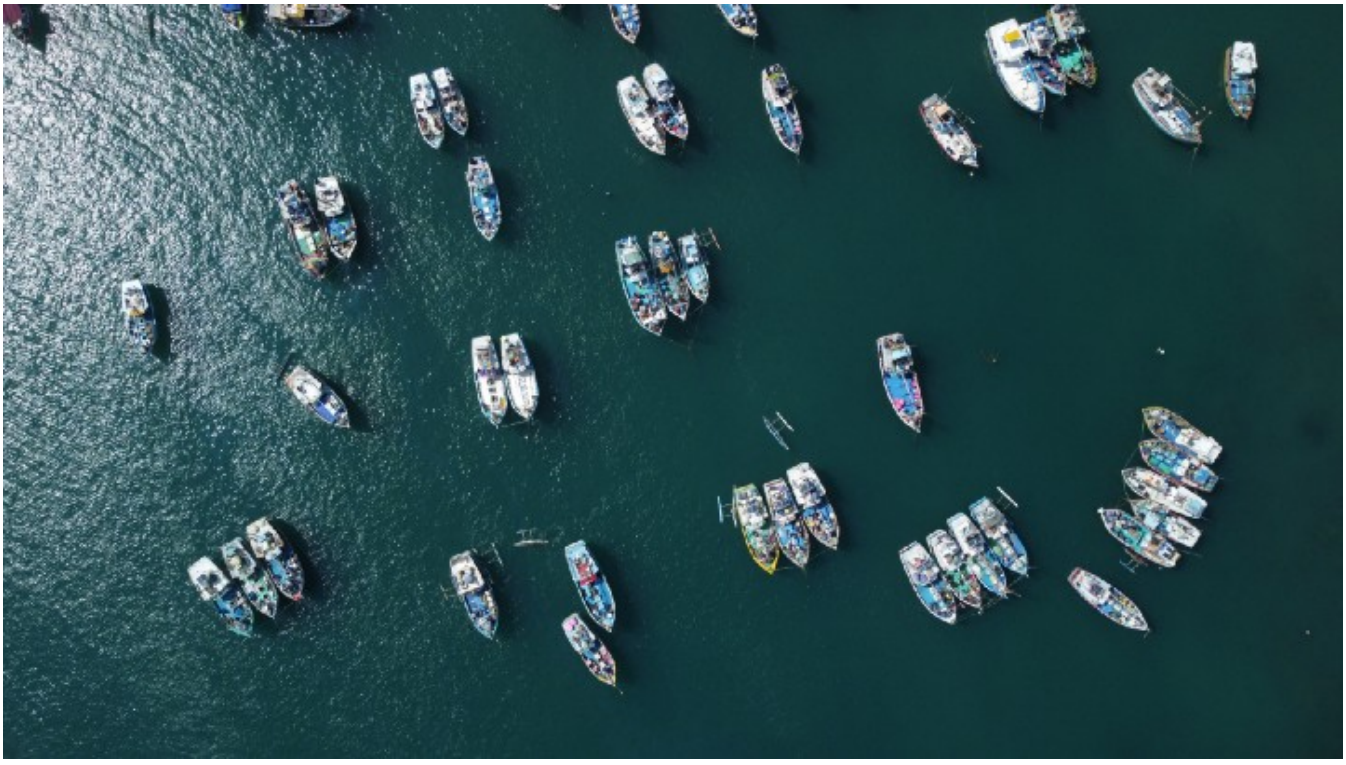


Photo by Ilya P on Unsplash

In helm, one chart can be dependent on another chart. For instance, a WordPress application requires a database to start functioning. In helm, we can deploy WordPress as part of the parent chart and MySQL or any other required application as a dependency of the parent chart.

Helm charts store their dependencies in **'charts/'.** There are two different ways to add dependency charts to parent charts:

1. Listing all the dependencies inside the **Chart.yaml** file and helm will download the dependencies and store them in the '**charts/**' directory**.**
2. Manually populate the dependency chart inside the '**charts/**' directory**.**

List dependencies inside the Chart.yaml file

Before listing dependencies in the **Chart.yaml** file, By default **Chart.yaml** file looks like this :

```
# Chart.yaml

apiVersion: v2
name: webserver
description: A Helm chart for Kubernetes
type: application
version: 0.1.0
appVersion: "1.16.0"
```

Now, let's see how we can populate the dependencies into the **Chart.yaml** file:

```
dependencies:
- name: mysql
  version: "9.3.4"
  repository: "https://charts.bitnami.com/bitnami"
```

After adding dependencies to the **Chart.yaml** file :

```
# Chart.yaml

apiVersion: v2
name: webserver
description: A Helm chart for Kubernetes
type: application
version: 0.1.0
appVersion: "1.16.0"

dependencies:
- name: mysql
```

```
  version: "9.3.4"
  repository: "https://charts.bitnami.com/bitnami"
```

Using a simple helm command we can pull the chart from the repository defined in the dependencies field:

```
   helm dependency update [CHART]
>> helm dependency update ~/webserver
```

The above-defined command will generate `.webserver/Chart.lock` file as well as download all dependencies into the `.webserver/charts` directory

The **`Chart.lock`** file lists the exact versions of immediate dependencies and their dependencies and so on.

Version Range

Instead of defining an exact version of a chart, we can define a version range. Helm allows us various ways to define version ranges.

we can view all the available versions of a chart using the following command:

```
   helm search repo [CHART] --versions
>> helm search repo  mysql  --versions
```

- **Hyphen Range Comparisons**

`version: 1.2 - 1.4.5` which are equivalent to `>= 1.2 <= 1.4.5`

```
dependencies:
- name: mysql
  version: 9.0.0 - 9.9.9
  repository: "https://charts.bitnami.com/bitnami"
```

Helm will download the latest MySQL chart and the version will be between `>= 9.0.0 <= 9.9.9` range. While writing this article the latest MySQL chart version is `9.4.1` . That's why helm will pull `9.4.1` version. The main advantage of defining the

version as a range is if a new version is available during the next update or installation of the parent chart. Helm will automatically download the latest version available within the specified range.

● Basic Comparisons

We can also use basic comparison operators for defining the version.

```
version: "= 9.4.1"
version: "!= 9.4.1"
version: "> 9.4.1"
version: "< 9.4.1"
version: ">= 9.4.1"
version: "<= 9.4.1"
version: ">= 9.4.1 and < 10.0.0"
```

● Caret(^) Range Comparisons (Major)

The caret (^) comparison operator is for major level changes once a stable (1.0.0) release has occurred.

```
^1.2.3  is equivalent to >= 1.2.3, < 2.0.0
^1.2.x  is equivalent to >= 1.2.0, < 2.0.0   # x as a placeholder
^2.3    is equivalent to >= 2.3, < 3
^2.x    is equivalent to >= 2.0.0, < 3       # x as a placeholder
^0      is equivalent to >= 0.0.0, < 1.0.0
```

● Tilde(~) Range Comparisons (Patch)

The tilde (~) comparison operator is for patch level ranges when a minor version is specified and major level changes when the minor number is missing. For example,

```
~1.3.2  is equivalent to >= 1.3.2, < 1.4.0
~3.4    is equivalent to >= 3.4, < 3.5
~3      is equivalent to >= 3, < 4
```

Enable or Disable dependencies

Depending on the user's needs a dependency can be enabled or disabled. If a dependency is disabled then during the installation of a chart the corresponding dependency will be ignored.

There are a couple of ways to achieve that. By using condition fields or tags fields.

●Using Condition

We can define a condition field while defining a dependency inside the **parentChart/Charts.yaml** file.

```
# parentChart/Charts.yaml

dependencies:
- name: mysql
  version: "9.3.4"
  repository: "https://charts.bitnami.com/bitnami"
  condition: mysql.enabled
```

The above-defined dependency will be installed only if the user sets `mysql.enabled` is **true** in the parent's **values.yaml** file.

```
# parentchart/values.yaml
...
...

mysql:
  enabled: false
```

As `mysql.enabled` is defined as **false**, then during the installation MySQL dependency will be ignored.

●Using Tags

Suppose there are multiple dependencies defined in the **Charts.yaml** file. To enable or disable it as per our needs, we have to do something similar to this :

```
# parentchart/Charts.yaml

dependencies:
- name: mysql
```

```
  version: "9.3.4"
  repository: https://charts.bitnami.com/bitnami
  condition: mysql.enabled

- name: mongodb
  version: 13.3.0
  repository: https://charts.bitnami.com/bitnami
  condition: mongodb.enabled
```

And the **values.yaml** file will look like the following demonstration:

```
# parentchart/values.yaml
....
....

mysql:
  enabled: true
mongodb
  enabled: false
```

If we use the `tags` field we can do it more simply. The tags field is a YAML list of labels to associate with a chart. **All charts with tags can be enabled or disabled by specifying the tag and a boolean value.** Let's see how we can do that :

First of all, let's see how we can define **tags** in the parent's **values.yaml** file:

```
# parentchart/values.yaml
...
...

tags:
  enabled: false
```

Delete the **`condtion`** field from the dependencies block of the parent's **Chart.yaml** file. Instead of **`condtion`** we will use **`tags`** at this moment.

```
# parentchart/Charts.yaml

dependencies:
- name: mysql
  version: "9.3.4"
  repository: https://charts.bitnami.com/bitnami
```

```
  - name: mongodb
    version: 13.3.0
    repository: https://charts.bitnami.com/bitnami
    tags:
      - enabled
```

In the above demonstration, we have defined **MongoDB** dependency with tags. As the values defined under the tags field is a boolean value and the value is defined as **false** in the parent's **values.yaml** file. As an outcome, **MongoDB** will be ignored when we install the parent chart. In contrast to MongoDB, **MySQL** dependency has no **`condtion`** or **`tags`** a defined. That is why **MySQL** will be installed as a dependency chart.

Listing dependencies

We can list all the dependencies for a given chart.

```
   helm dependency list  [CHART]
>> helm dependency list  ~/webserver/

-----------------------------------------------------------
NAME    VERSION REPOSITORY                                STATUS
mysql   9.3.4   https://charts.bitnami.com/bitnami       ok
mongodb 2.0.3   https://charts.bitnami.com/bitnami       missing
```

In the above demonstration, we can see that the status is "ok" for MySQL and "missing" for MongoDB. The reason for that is MySQL chart is already pulled from the repository and stored in the '**chart/**' directory. While the MongoDB chart is not available. By executing the **dependency update** command all the missing dependencies can be pulled and stored.

> Thank You 🖤

All Articles on Helm Chart -

Helm — In Action
Helm — Advanced Commands
Helm — Create Your First Chart
Helm — Template Actions, Functions, and Pipelines
Helm — Flow Control
Helm — Variables
Helm — Named Template
Helm — Dependencies

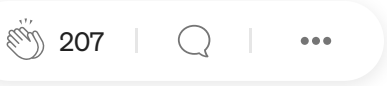
Level Up Coding

Thanks for being a part of our community! Before you go:

- 👋 Clap for the story and follow the author 👉
- 📰 View more content in the Level Up Coding publication
- 🔔 Follow us: Twitter | LinkedIn | Newsletter

🚀👉 **Join the Level Up talent collective and find an amazing job**

Helm Helm Chart Kubernetes Kubernetes Cluster Package Manager



Enjoy the read? Reward the writer.[Beta]

Your tip will go to Md Shamim through a third-party platform of their choice, letting them know you appreciate their story.

Give a tip