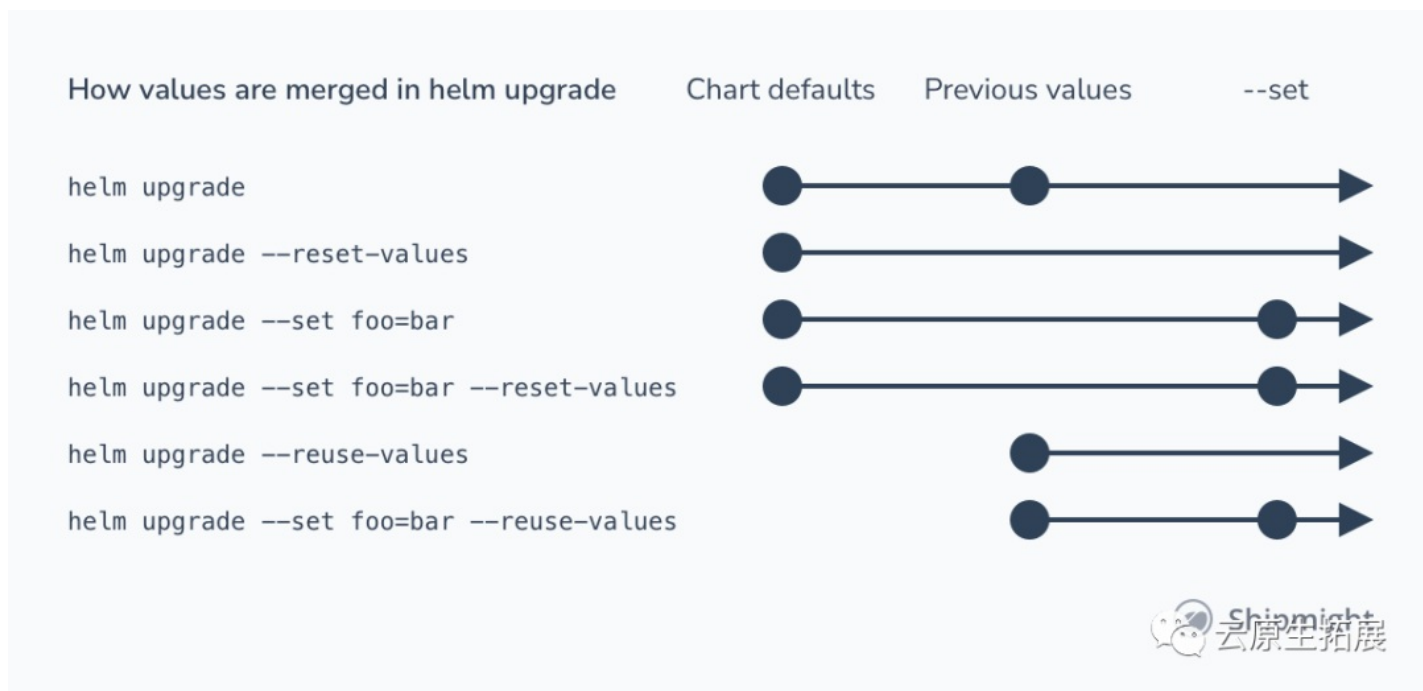


Kubernetes 系列（三十六）理解 Helm upgrade 一些重要参数

在运行 `helm upgrade` 时，您需要时不时地使用 `--reset-values` 和 `--reuse-values` 标志。让我们深入了解它们的实际工作原理，并看看 chart 的 values 在升级之间发生了变化时的一个问题。

以下是表格中可视化的不同场景：



介绍

Helm 的要点在于，您可以自定义 Chart 的参数(在Helm的行话中，参数被称为 values)。

例如，作为loki-stack Chart 的一部分，您可以通过 `--set` 设置一个值来启用和禁用堆栈的组件：

```
helm install example-loki grafana/loki-stack --set grafana.enabled=true
```

这同样适用于升级现有 release。您可以更改一个值，chart 将根据您的自定义重新配置：

```
helm upgrade example-loki grafana/loki-stack --set grafana.enabled=false
```

注意到，即使您只是想使用已经安装的同 Chart 更新值，也必须指定 chart 名称？这是因为upgrade-命令还能够将发布版本升级到 Chart 的不同版本(技术上也可以升级到任何其他 Chart，尽管这听起来不像一个常见的用例)。一个常见的工作流程是从远程回购更新 Chart，然后将发布版本升级到最新版本：

```
helm repo update
helm upgrade ...
```

也可以设置一个特定版本的 chart 来升级：

```
helm upgrade example-loki grafana/loki-stack --version 1.2.3
```

事实上，您可以重新配置和更新 Chart 是Helm的全部目的。我敢肯定，如果你之前使用过Helm，你对上面所有的东西都很熟悉。

但是配置更改合并在一起的方式可能会让您猝不及防。upgrade-命令包含一些内部逻辑，可能看起来不一致。如果要升级的 Chart 的值与最初的Chart 版本相比发生了变化，那么情况可能会更加混乱。我个人不得不逐个测试不同的变体，以完全理解所有的内容。

第一个潜在的惊喜是合并行为的差异实际上是由这个决定的:您也在升级期间设置值吗?

如果你是，helm 含蓄地使用了“reset values”策略。

如果你不是，helm 含蓄地使用“reuse values”策略。

这两种策略都可以通过CLI标志来实现，我们将在下面介绍。

最后，当 Chart 中的值模式发生更改时，还有一个与重用值有关的重要问题需要了解。最后我也会讲到这个。

默认行为

让我们以例子的形式来复习一下。

当你简单地将 chart 升级到一个新版本时，你之前的值将保持有效:

example-chart 1.0.0

```
// values.yaml
image:
  name: nginx
  pullPolicy: Always
tag: ""
```

Previously set values

```
image:
  pullPolicy: Never
```

Final values applied to chart

```
image:
  name: nginx
  pullPolicy: Never
tag: ""
```

example-chart 1.2.0

```
// values.yaml
image:
  name: nginx
  pullPolicy: Always
tag: 1.2.0
```

Updated values from CLI

None

Final values applied to chart

```
image:
  name: nginx
  pullPolicy: Never
tag: 1.2.0
```

```
$ helm upgrade foobar example-chart --version 1.2.0
```

Implicit --reuse-values -like behaviour when not using -f / --set / --set-string!



这就是我前提到的“reuse values”策略。

但是如果您在升级期间也设置了一些值，那么您以前的值实际上会被删除!

example-chart 1.0.0

```
// values.yaml
image:
  name: nginx
  pullPolicy: Always
  tag: ""
```

Previously set values

```
image:
  pullPolicy: Never
```

Final values applied to chart

```
image:
  name: nginx
  pullPolicy: Never
  tag: ""
```

example-chart 1.2.0

```
// values.yaml
image:
  name: nginx
  pullPolicy: Always
  tag: 1.2.0
```

Updated values from CLI

```
image:
  name: foo
```

Final values applied to chart

```
image:
  name: foo
  pullPolicy: Always
  tag: 1.2.0
```

Base values are
reset to chart
defaults.

```
$ helm upgrade ... example-chart --version 1.2.0 --set image.name=foo
```

Implicit --reset-values when using -f / --set / --set-string!



这就是我前提到的“reset values”策略。值将重置为新 chart 版本的默认值，您在CLI上指定的任何新值都将在此基础上合并。

这是相当令人惊讶的!您需要小心，不要不小心撤销一些定制。

显式指定行为

您可以通过CLI标志来控制使用哪种策略。

如果你没有设置新的值，但想重置为 chart 默认值，你可以通过 `--reset-values` 来实现:

example-chart 1.0.0

```
// values.yaml
image:
  name: nginx
  pullPolicy: Always
  tag: ""
```

Previously set values

```
image:
  pullPolicy: Never
```

Final values applied to chart

```
image:
  name: nginx
  pullPolicy: Never
  tag: ""
```

example-chart 1.2.0

```
// values.yaml
image:
  name: nginx
  pullPolicy: Always
  tag: 1.2.0
```

Updated values from CLI

None

Final values applied to chart

```
image:
  name: nginx
  pullPolicy: Always
  tag: 1.2.0
```

Base values are reset to chart defaults.

```
$ helm upgrade foobar example-chart --version 1.2.0 --reset-values
```

Note that you can still set new values via --set and friends while using this flag. They will be merged to the original chart values.



在相反的情况下，如果你正在设置新的值，但想要将它们合并到以前的值，你可以使用 `--reuse-values`：

example-chart 1.0.0

```
// values.yaml
image:
  name: nginx
  pullPolicy: Always
  tag: ""
```

Previously set values

```
image:
  pullPolicy: Never
```

Final values applied to chart

```
image:
  name: nginx
  pullPolicy: Never
  tag: ""
```

example-chart 1.2.0

```
// values.yaml
image:
  name: nginx
  pullPolicy: Always
  tag: 1.2.0
```

Updated values from CLI

```
image:
  name: foo
```

Final values applied to chart

```
image:
  name: foo
  pullPolicy: Never
  tag: 1.2.0
```

```
$ helm upgrade ... --version 1.2.0 --set image.name=foo --reuse-values
```

New values are merged to values from previous release. See the gotcha below!



很简单，对吧？

只要新 chart 版本中的值模式没有改变，这就可以工作。所以，大部分情况下，一切正常！但如果模式改变了呢？

使用--reuse-values 以及 模式改变

有时更新后的 chart 的 `values.yaml` 值会发生变化。例如，可能添加了一些部分，或者对现有的部分做了一些更改。您希望这些更改合并到您的新版本中，对吗？

他们就是这样！但只有在不使用 `--reuse-values` 的情况下。这是令人困惑的，因为它使标志的行为不同于我们的第一个例子，它的工作方式几乎是 `--reuse-values`，但也合并来自 chart 的更改。

因此，澄清一下。当你进行一个简单的升级而不设置任何值或标志时，更新的 `values.yaml` 从新的 Chart 版本被合并到你的版本中，正如预期的那样：

example-chart 1.0.0

```
// values.yaml
image:
  name: nginx
  pullPolicy: Always
tag: ""
```

Previously set values

```
image:
  pullPolicy: Never
```

Final values applied to chart

```
image:
  name: nginx
  pullPolicy: Never
tag: ""
```

example-chart 1.2.0

```
// values.yaml
image:
  name: nginx
  pullPolicy: Always
thisIsNew: true
```

Updated values from CLI

None

Final values applied to chart

```
image:
  name: nginx
  pullPolicy: Never
thisIsNew: true
```

Schema changes from new version were merged in as expected.

```
$ helm upgrade foobar example-chart --version 1.2.0
```



但当你想同时更新之前设置的值时，你自然会使用 `--reuse-values` 以及 `--set`，对吧？在本例中，`--reuse-values` 有一个意想不到的效果。这导致 Helm 从你之前的版本中“reuse values”作为基础，而忽略了新 chart 版本中可能发生的任何变化：

example-chart 1.0.0

```
// values.yaml
image:
  name: nginx
  pullPolicy: Always
  tag: ""
```

Previously set values

```
image:
  pullPolicy: Never
```

Final values applied to chart

```
image:
  name: nginx
  pullPolicy: Never
  tag: ""
```

example-chart 1.2.0

```
// values.yaml
image:
  name: nginx
  pullPolicy: Always
  thisIsNew: true
```

Updated values from CLI

```
image:
  name: foo
```

Final values applied to chart

```
image:
  name: foo
  pullPolicy: Never
  tag: ""
```

Huh?
Helm just reused
old release values
as expected, but
ignored changes
in new chart
version.

```
$ helm upgrade ... --version 1.2.0 --set image.name=foo --reuse-values
```

Workaround: `helm get values > ... && helm upgrade -f ...`



这不是你想的逻辑，对吧？当然，您确实显式地要求重用值，但是为什么不希望它们首先与图表中的(更新的)默认值合并呢？特别是因为这是升级的常见行为。

至少对我来说，这是非常不直观的，因为我希望包管理器使用 chart 值作为基础，然后在其上合并我以前的发布值，最后合并我在升级中提供的任何附加值。

当这种情况发生时，您的升级可能会失败，因为模板可能会因过时的值结构而失败。例如，如果一个新的配置部分被添加到 chart 中，但你的发布值实际上缺少整个部分，你肯定会遇到一些 nil 指针错误。

为什么是这样的？引用 GitHub 的一位维护者的话：“设计目标是防止新 chart 版本的值的更改被自动应用。”意思是，这是有意的(不是 bug)，这些标志的行为不会被改变。

这仍然非常令人困惑，因为这些看似相似的命令之间的行为非常不直观，可能导致升级中断。

至于替代方案，GitHub 从 2020 年 5 月开始有一个活跃的问题，讨论了变通方案和新功能的可能性。一个新的标志，如 `--reset-then-reuse-values` 可能是未来添加到 helm 澄清这个问题。

同时，如果你想在 chart 值可能发生变化时获得预期的 `--reuse-values` 行为，你应该先将旧值转储到一个文件中，然后使用它作为没有 `--reuse-values` 的基础：

```
helm get values example-loki > prev-values.yaml
helm upgrade example-loki -f prev-values.yaml --set grafana.enabled=true
```

通过这种方式，您以前的值将被合并到更新的 chart 值中，最后来自 CLI 的任何附加值都将在此基础上合并。

总结

了解这些标志是如何工作的肯定会帮助您在运行 helm 升级时减少调试时间。我希望这篇文章在说明它们之间的区别方面对您有所帮助。

我试着把这些信息揉进这个表格：

How values are merged in helm upgrade

Chart defaults

Previous values

--set

helm upgrade

helm upgrade --reset-values

helm upgrade --set foo=bar

helm upgrade --set foo=bar --reset-values

helm upgrade --reuse-values

helm upgrade --set foo=bar --reuse-values



If you ever need to figure out a specific scenario, I recommend creating two local charts via `helm create test-chart-1` and `helm create test-chart-2` and upgrading between them, e.g. `helm install test test-chart-1 --set foo=bar && helm upgrade test test-chart-2 --reset-values && helm get values --all test`.

如果你需要想出一个特定的场景，我建议通过 `helm create test-chart-1` 和 `helm create test-chart-2` 创建两个本地 chart，并在它们之间升级，例如 `helm install test test-chart-1 --set foo=bar && helm upgrade test test-chart-2 --reset-values && helm get values --all test`.

我仍将这篇文章中的所有信息图表收集到一个大的图像中，所以如果这有助于注意到差异，你可以在它周围平移和缩放：

Helm either does or doesn't merge new values into values from your previous release. It depends on whether or not you also update values while upgrading.

example-chart 1.0.0	example-chart 1.2.0
<pre>// values.yaml image: name: nginx pullPolicy: Always tag: ""</pre>	<pre>// values.yaml image: name: nginx pullPolicy: Always tag: 1.2.0</pre>
Previously set values	Updated values from CLI
<pre>image: pullPolicy: Never</pre>	None
Final values applied to chart	Final values applied to chart
<pre>image: name: nginx pullPolicy: Never tag: ""</pre>	<pre>image: name: nginx pullPolicy: Never tag: 1.2.0</pre>
<pre>\$ helm upgrade foobar example-chart --version 1.2.0</pre>	
Implicit --reuse-values-like behaviour when not using -f/--set/--set-string! 	

example-chart 1.0.0	example-chart 1.2.0
<pre>// values.yaml image: name: nginx pullPolicy: Always tag: ""</pre>	<pre>// values.yaml image: name: nginx pullPolicy: Always tag: 1.2.0</pre>
Previously set values	Updated values from CLI
<pre>image: pullPolicy: Never</pre>	<pre>image: name: foo</pre>
Final values applied to chart	Final values applied to chart
<pre>image: name: nginx pullPolicy: Never tag: ""</pre>	<pre>image: name: foo pullPolicy: Always tag: 1.2.0</pre>
<pre>\$ helm upgrade _ example-chart --version 1.2.0 --set image.name=foo</pre>	
Implicit --reset-values when using -f/--set/--set-string! 	

Base values are reset to chart defaults.


You can set explicit behaviour with the --reset-values and --reuse-values flags.

example-chart 1.0.0	example-chart 1.2.0
<pre>// values.yaml image: name: nginx pullPolicy: Always tag: ""</pre>	<pre>// values.yaml image: name: nginx pullPolicy: Always tag: 1.2.0</pre>
Previously set values	Updated values from CLI
<pre>image: pullPolicy: Never</pre>	None
Final values applied to chart	Final values applied to chart
<pre>image: name: nginx pullPolicy: Never tag: ""</pre>	<pre>image: name: nginx pullPolicy: Always tag: 1.2.0</pre>
<pre>\$ helm upgrade foobar example-chart --version 1.2.0 --reset-values</pre>	
Note that you can still set new values via --set and friends while using this flag. They will be merged to the original chart values. 	

Base values are reset to chart defaults.

example-chart 1.0.0	example-chart 1.2.0
<pre>// values.yaml image: name: nginx pullPolicy: Always tag: ""</pre>	<pre>// values.yaml image: name: nginx pullPolicy: Always tag: 1.2.0</pre>
Previously set values	Updated values from CLI
<pre>image: pullPolicy: Never</pre>	<pre>image: name: foo</pre>
Final values applied to chart	Final values applied to chart
<pre>image: name: nginx pullPolicy: Never tag: ""</pre>	<pre>image: name: foo pullPolicy: Never tag: 1.2.0</pre>
<pre>\$ helm upgrade _ --version 1.2.0 --set image.name=foo --reuse-values</pre>	
New values are merged to values from previous release. See the gotcha below! 	

However! Note the gotcha with --reuse-values if the values schema has changed in the new chart version

example-chart 1.0.0	example-chart 1.2.0
<pre>// values.yaml image: name: nginx pullPolicy: Always tag: ""</pre>	<pre>// values.yaml image: name: nginx pullPolicy: Always thisIsNew: true</pre>
Previously set values	Updated values from CLI
<pre>image: pullPolicy: Never</pre>	None
Final values applied to chart	Final values applied to chart
<pre>image: name: nginx pullPolicy: Never tag: ""</pre>	<pre>image: name: nginx pullPolicy: Never thisIsNew: true</pre>
<pre>\$ helm upgrade foobar example-chart --version 1.2.0</pre>	
	

Schema changes from new version were merged in as expected.

example-chart 1.0.0	example-chart 1.2.0
<pre>// values.yaml image: name: nginx pullPolicy: Always tag: ""</pre>	<pre>// values.yaml image: name: nginx pullPolicy: Always thisIsNew: true</pre>
Previously set values	Updated values from CLI
<pre>image: pullPolicy: Never</pre>	<pre>image: name: foo</pre>
Final values applied to chart	Final values applied to chart
<pre>image: name: nginx pullPolicy: Never tag: ""</pre>	<pre>image: name: foo pullPolicy: Never tag: ""</pre>
<pre>\$ helm upgrade _ --version 1.2.0 --set image.name=foo --reuse-values</pre>	
Workaround: <code>helm get values > _ && helm upgrade</code>  云原生拓展	

Huh? Helm just reused old release values as expected, but ignored changes in new chart version.

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。