链滴

奇妙的 Docker Inspect 模版

作者：88250

原文链接：https://ld246.com/article/1427784659823

来源网站：链滴

<h3>概述</h3>
很多 Docker 用户都知道 docker inspect 命令，该命令用于获取容器/镜像的元据，其中 -f 参数可以用于获取指定的数据，例如使用 docker inspect -f {{.IPAddress}} 来获取容器的 IP 地址。不过很多用户容易被该特性的语法搞晕，并很少人能将它的优势发挥出来（大部分人都是通过 grep 来获取指定数据，虽然有但比较零散混乱）。本文将详细介绍 -f 参数，并给出一些例子来说明如使用它。
<h3>docker inspect </h3>
简单地说，-f 的实参是个 Go 模版，并在容器/镜像的元数据上以该 Go 模作为输入，最终返回模版指定的数据。第一个问题就是该命令说明文档中的用词“Go 模版”对于没有 Go 开发经验的人来说很模糊——我的第一感觉就是它类似恐怖的 C++ 模版。还好 Go 模版和 C++ 版/泛型毫不相干，Go 模版是一种模板引擎，让数据以指定的模式输出。这个概念对于 Web 开发者是非常熟悉，Web 领域有很多模版引擎，比如 Jinga2（用于 Python 和 Flask）、Mustache、JSP 等等，看下的简单示例：
<pre class="prettyprint">$ docker inspect -f 'Hello from container {{.Name}}' jenkins
Hello from container /jenkins</pre>
我们可以看到，-f 指定了一个简单的模式（或称之为模版）并应用于容器的元数据。当然，对于元数据，我们也可以什么都不指定：
<pre class="prettyprint">$ docker inspect -f "This is a bit pointless" jenkins
This is a bit pointless</pre>
下面让我们来进一步看看 Go 模版的奇妙之处，例如我们可以通过模版来查找有退出码为非 0 的容器名：
<pre class="prettyprint">$ docker inspect -f '{{if ne 0.0 .State.ExitCode }}{{.Name}} {{.State.Exitode}}{{ end }}' $(docker ps -aq)

/tender_colden 1

/clever_mcclintock 126

/grave_bartik 1</pre>

（无论是否匹配到，对于每个容器都会输出一行）
或者在 jenkins-data 容器中查找卷 /var/jenkins_home 对应在 host 的目录：
<pre class="prettyprint">docker inspect -f '{{index .Volumes "/var/jenkins_home"}}' jenkins-data
/var/lib/docker/vfs/dir/5a6f7b306b96af38723fc4d31def1cc515a0d75c785f3462482f60b73053b1a</pre>
上面的例子可能稍微有点难理解，不过没关系，我们先来了解一下 Go 模版的基本用法
<h3>模版指令</h3>
{{ }} 语法用于处理模版指令，大括号外的任何字符都将直接输出。
<h3>上下文</h3>
“.” 表示“当前上下文”。大多数情况下表示了容器元数据的整个数据结构，但在某些情况下以重新规定上下文，比如使用 with 函数：
<pre class="prettyprint">$ docker inspect -f '{{.State.Pid}}' jenkins
6331
$ docker inspect -f '{{with .State}} {{.Pid}} {{end}}' jenkins
6331</pre>
可以使用 $ 来获取根上下文，例如：
<pre class="prettyprint">$ docker inspect -f '{{with .State}} {{$.Name}} has pid {{.Pid}} {{end}}' enkins
 /jenkins has pid 6331</pre>

注意，单独使用 “.” 本身也是可以的，将输出未格式化的完整元数据：</spa
>
<pre class="prettyprint">$ docker inspect -f '{{.}}' jenkins
...</pre>
<h3>数据类型</h3>
inspect 数据可以由浮点数、字符串和布尔组成，可以使用 Go 模版内置函数进行比较判断。虽然 Go 模版支持整数，但目前 inspect 数据中的数值类型都是浮点数，而整数应该对于大多数场景更方（详见该 <a href="https://github.com/docker/docker/issues/11641" target="_blank">Issue</
>）。使用字符串时可以使用双引号。
数据中不存在的值是不可以用来比较的：
<pre class="prettyprint">$ docker inspect -f '{{.ExecIDs}}' jenkins
<no value>
$ docker inspect -f '{{eq .ExecIDs .ExecIDs}}' jenkins
FATA[0000] template: :1:2: executing "" at <eq .ExecIDs .ExecIDs>: error calli
g eq: invalid type for comparison</pre>
<h3>数据结构</h3>
inspect 数据使用 map 以及数组保存。Map 结构非常简单，前面我们曾经展示过，可以通过 . 链式来访问 map 内部数据：
<pre class="prettyprint">$ docker inspect -f '{{.State.ExitCode}}' jenkins
0</pre>
不过有些情况（比如 map 的键不是字符串）是不能直接使用 . 方式来获取 map 值的，此时我可以使用 index 函数，前面卷的例子可以这样写：
<pre class="prettyprint">docker inspect -f '{{index .Volumes "/var/jenkins_home"
}}' jenkins-data
/var/lib/docker/vfs/dir/5a6f7b306b96af38723fc4d31def1cc515a0d75c785f3462482f60b73053
b1a</pre>
我们也可以使用 index 来获取指定下标的数组值：
/span>
<pre class="prettyprint">$ docker inspect -f '{{.HostConfig.Binds}}' jenkins
[/var/run/docker.sock:/var/run/docker.sock /usr/bin/docker:/usr/bin/docker]
$ docker inspect -f '{{index .HostConfig.Binds 1}}' jenkins
/usr/bin/docker:/usr/bin/docker</pre>
<h3>函数</h3>
除了 index 函数，其他很多函数也很常用。比如逻辑函数 a
d、or 可以返回布尔结果。注意，函数是不能放在中间：
<pre class="prettyprint">$ docker inspect -f '{{and true true}}' jenkins
true</pre>
而不是：
<pre class="prettyprint">$ docker inspect -f '{{true and true}}' jenkins
FATA[0000] template: :1:2: executing "" at <true>: can't give argument to n
n-function true </pre>
下面是一些常用的比较函数：

eq (等于)
ne (不等于)
lt (小于)
le (小于等于)
gt (大于)
ge (大于等于)

我们可以用这些函数来比较字符串、浮点数或整数：
<pre class="prettyprint">$ docker inspect -f '{{eq "abc" "abc"}}' jenki
s
true

```
$ docker inspect -f '{{ge 1 -1}}' jenkins
true
$ docker inspect -f '{{lt 4.5 4.6}}' jenkins
true
$ docker inspect -f '{{ne 4.5 4.5}}' jenkins
false</pre>
<p><span><span>要注意的是操作数类型必须匹配，数字比较时使用浮点数：</span></span></p>
<pre class="prettyprint">$ docker inspect -f '{{eq &quot;4.5&quot; 4.5}}' jenkins
FATA[0000] template: :1:2: executing &quot;&quot; at &lt;eq &quot;4.5&quot; 4.5&gt;: error
alling eq: incompatible types for comparison
$ docker inspect -f '{{gt .State.Pid 1}}' jenkins
FATA[0000] template: :1:2: executing &quot;&quot; at &lt;gt .State.Pid 1&gt;: error calling gt:
ncompatible types for comparison
$ docker inspect -f '{{gt .State.Pid 1.0}}' jenkins
true</pre>
<p>另外，可以使用 <em>json</em> 函数来生成 JSON 输出：</p>
<pre class="prettyprint">$ docker inspect -f '{{json .NetworkSettings.Ports}}' jenkins
{&quot;50000/tcp&quot;:null,&quot;8080/tcp&quot;:[{&quot;HostIp&quot;:&quot;0.0.0.0&q
ot;,&quot;HostPort&quot;:&quot;8080&quot;}]}</pre>
<p>我们也可以使用 <a href="https://stedolan.github.io/jq/">jq</a> 工具来组合果：</p>
<pre class="prettyprint">$ docker inspect -f '{{json .State}}' jenkins-data | jq '.StartedAt'
&quot;2015-03-15T20:26:30.526796706Z&quot;</pre>
<p><span><span>当然，<em>docker inspect</em> 的默认输出结果就是 JSON，所以面这样也可以：</span></span></p>
<pre class="prettyprint">$ docker inspect jenkins-data | jq '.[] | .State.StartedAt'
&quot;2015-03-15T20:26:30.526796706Z&quot;</pre>
<p>更多函数请参考 <a href="http://golang.org/pkg/text/template/" target="_blank">Go 官文档</a>，不过很奇怪的是官方文档中并没有描述 <em>json</em> 函数（我是从 <a href
"http://nathanleclaire.com/blog/2014/07/12/10-docker-tips-and-tricks-that-will-make-you-si
g-a-whale-song-of-joy/">Nathan LeClaire’s blog</a> 中学到的），你如果知道其中原，记得告诉我！</p>
<h3>If 语句</h3>
<p>条件语句 <em>if</em> 可以和前面的比较函数一起使用：</p>
<pre class="prettyprint">$ docker inspect -f '{{if eq .State.ExitCode 0.0}}
Normal Exit
{{else if eq .State.ExitCode 1.0}}
Not a Normal Exit
{{else}}
Still Not a Normal Exit
{{end}}' jenkins

Normal Exit</pre>

<p>注意，<em>{{end}}</em> 语句必须有，<em>else if</em> 和 <em>else</em> 需使用。</p>
<h3>结论</h3>
<p>我想本文应该涵盖了 <em>docker inspect -f</em> 使用模版时的大部分内容，不过另还有一些很常用的特性，比如使用 <em>range</em> 来迭代数据、自定义函数、使管道等需要你来自己摸索实践。</p>
<p>我还没找到一份全面的使用 Go 模版的参考文档，目前我觉得这份还不错：<a href="http://jan.ewmarch.name/go/template/chapter-template.html">chapter from a the free e-book “Netwrk programming with Go”</a> by Jan Newmarch。</p>
```

当然，你可以参考 Go&bsp;官方文档，但是它太精简了，特别是对于非 Go 程序员来说比较难理解。

本文译自：Docker Inspect Template Magic