对于没有使用过 kubernetes 的 docker 用户，如何快速掌握 kubectl 命令？在本文中，我们将向 docker-cli 用户介绍 Kubernetes 命令行如何与 api 进行交互。该命令行工具—kubectl，被设计成 docker-cli 用户所熟悉的样子，但是它们之间又存在一些必要的差异。该文档将向您展示每个 docker 子命令和 kubectl 与其等效的命令。

在使用 kubernetes 集群的时候，docker 命令通常情况是不需要用到的，只有在调试程序或者容器的时候用到，我们基本上使用 kubectl 命令即可，所以在操作 kubernetes 的时候我们抛弃原先使用 docker 时的一些观念。

docker run

如何运行一个 nginx Deployment 并将其暴露出来？ 查看 kubectl run 。

使用 docker 命令：

```
$ docker run -d --restart=always -e DOMAIN=cluster --name
nginx-app -p 80:80 nginx
a9ec34d9878748d2f33dc20cb25c714ff21da8d40558b45bfaec9955859075d0
$ docker ps
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS
PORTS                           NAMES
a9ec34d98787        nginx               "nginx -g 'daemon of
2 seconds ago       Up 2 seconds        0.0.0.0:80->80/tcp,
443/tcp   nginx-app
```

使用 kubectl 命令：

```
# start the pod running nginx
$ kubectl run --image=nginx nginx-app --port=80 --
env="DOMAIN=cluster"
deployment "nginx-app" created
```

在大于等于 1.2 版本 Kubernetes 集群中，使用kubectl run 命令将创建一个名为 "nginx-app" 的 Deployment。如果您运行的是老版本，将会创建一个 replication controller。 如果您想沿用旧的行为，使用 --generation=run/v1 参数，这样就会创建 replication controller。查看 kubectl run 获取更多详细信息。

```
# expose a port through with a service
$ kubectl expose deployment nginx-app --port=80 --name=nginx-http
service "nginx-http" exposed
```

在 kubectl 命令中，我们创建了一个 Deployment，这将保证有 N 个运行 nginx 的 pod（N 代表 spec 中声明的 replica 数，默认为 1）。我们还创建了一个 service，使用 selector 匹配具有相应的 selector 的 Deployment。查看 快速开始 获取更多信息。
默认情况下镜像会在后台运行，与docker run -d ... 类似，如果您想在前台运行，使用：

```
kubectl run [-i] [--tty] --attach <name> --image=<image>
```

与 docker run ... 不同的是，如果指定了 --attach ，我们将连接到 stdin，stdout 和 stderr，而不能控制具体连接到哪个输出流（docker -a ...）。
因为我们使用 Deployment 启动了容器，如果您终止了连接到的进程（例如 ctrl-c），容器将会重启，这跟 docker run -it不同。 如果想销毁该 Deployment（和它的 pod），您需要运行 kubeclt delete deployment <name>。

docker ps

如何列出哪些正在运行？查看 kubectl get。
使用 docker 命令：

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS
PORTS                           NAMES
a9ec34d98787        nginx               "nginx -g 'daemon of
About an hour ago   Up About an hour    0.0.0.0:80->80/tcp,
443/tcp   nginx-app
```

使用 kubectl 命令：

```
$ kubectl get po
NAME              READY     STATUS    RESTARTS   AGE
nginx-app-5jyvm   1/1       Running   0          1h
```

docker attach

如何连接到已经运行在容器中的进程？查看 kubectl attach。

使用 docker 命令：

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS
PORTS                           NAMES
a9ec34d98787        nginx               "nginx -g 'daemon of
8 minutes ago       Up 8 minutes        0.0.0.0:80->80/tcp,
443/tcp   nginx-app
$ docker attach a9ec34d98787
...
```

使用 kubectl 命令：

```
$ kubectl get pods
NAME              READY     STATUS    RESTARTS   AGE
nginx-app-5jyvm   1/1       Running   0          10m
$ kubectl attach -it nginx-app-5jyvm
...
```

docker exec

如何在容器中执行命令？查看 kubectl exec。

使用 docker 命令：

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS
PORTS                           NAMES
a9ec34d98787        nginx               "nginx -g 'daemon of
8 minutes ago       Up 8 minutes        0.0.0.0:80->80/tcp,
443/tcp   nginx-app
$ docker exec a9ec34d98787 cat /etc/hostname
a9ec34d98787
```

使用 kubectl 命令：

```
$ kubectl get po
NAME              READY     STATUS    RESTARTS   AGE
nginx-app-5jyvm   1/1       Running   0          10m
$ kubectl exec nginx-app-5jyvm -- cat /etc/hostname
nginx-app-5jyvm
```

执行交互式命令怎么办？

使用 docker 命令：

```
$ docker exec -ti a9ec34d98787 /bin/sh
# exit
```

使用 kubectl 命令：

```
$ kubectl exec -ti nginx-app-5jyvm -- /bin/sh
# exit
```

更多信息请查看 获取运行中容器的 Shell 环境。

docker logs

如何查看运行中进程的 `stdout/stderr`？查看 `kubectl logs`。

使用 `docker` 命令：

```
$ docker logs -f a9e
192.168.9.1 - - [14/Jul/2015:01:04:02 +0000] "GET / HTTP/1.1"
200 612 "-" "curl/7.35.0" "-"
192.168.9.1 - - [14/Jul/2015:01:04:03 +0000] "GET / HTTP/1.1"
200 612 "-" "curl/7.35.0" "-"
```

使用 `kubectl` 命令：

```
$ kubectl logs -f nginx-app-zibvs
10.240.63.110 - - [14/Jul/2015:01:09:01 +0000] "GET / HTTP/1.1"
200 612 "-" "curl/7.26.0" "-"
10.240.63.110 - - [14/Jul/2015:01:09:02 +0000] "GET / HTTP/1.1"
200 612 "-" "curl/7.26.0" "-"
```

现在是时候提一下 `pod` 和容器之间的细微差别了；默认情况下如果 `pod` 中的进程退出 `pod` 也不会终止，相反它将会重启该进程。这类似于 `docker run` 时的 `--restart=always` 选项， 这是主要差别。在 `docker` 中，进程的每个调用的输出都是被连接起来的，但是对于 `kubernetes`，每个调用都是分开的。要查看以前在 `kubernetes` 中执行的输出，请执行以下操作：

```
$ kubectl logs --previous nginx-app-zibvs
10.240.63.110 - - [14/Jul/2015:01:09:01 +0000] "GET / HTTP/1.1"
200 612 "-" "curl/7.26.0" "-"
10.240.63.110 - - [14/Jul/2015:01:09:02 +0000] "GET / HTTP/1.1"
200 612 "-" "curl/7.26.0" "-"
```

查看 记录和监控集群活动 获取更多信息。

docker stop 和 docker rm

如何停止和删除运行中的进程？查看 kubectl delete。

使用 docker 命令：

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS
PORTS                         NAMES
a9ec34d98787        nginx               "nginx -g 'daemon of
22 hours ago        Up 22 hours         0.0.0.0:80->80/tcp,
443/tcp   nginx-app
$ docker stop a9ec34d98787
a9ec34d98787
$ docker rm a9ec34d98787
a9ec34d98787
```

使用 kubectl 命令：

```
$ kubectl get deployment nginx-app
NAME         DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx-app    1         1         1            1           2m

$ kubectl get po -l run=nginx-app
NAME                         READY     STATUS    RESTARTS   AGE
nginx-app-2883164633-aklf7   1/1       Running   0          2m

$ kubectl delete deployment nginx-app
deployment "nginx-app" deleted

$ kubectl get po -l run=nginx-app
# Return nothing
```

请注意，我们不直接删除 pod。使用 kubectl 命令，我们要删除拥有该 pod 的 Deployment。如果我们直接删除pod，Deployment 将会重新创建该 pod。

docker version

如何查看客户端和服务端的版本？查看 kubectl version。

使用 docker 命令：

```
$ docker version
Client version: 1.7.0
Client API version: 1.19
Go version (client): go1.4.2
Git commit (client): 0baf609
OS/Arch (client): linux/amd64
Server version: 1.7.0
Server API version: 1.19
Go version (server): go1.4.2
Git commit (server): 0baf609
OS/Arch (server): linux/amd64
```

使用 kubectl 命令：

```
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"6",
GitVersion:"v1.6.9+a3d1dfa6f4335",
GitCommit:"9b77fed11a9843ce3780f70dd251e92901c43072",
GitTreeState:"dirty", BuildDate:"2017-08-29T20:32:58Z",
OpenPaasKubernetesVersion:"v1.03.02", GoVersion:"go1.7.5",
Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"6",
GitVersion:"v1.6.9+a3d1dfa6f4335",
GitCommit:"9b77fed11a9843ce3780f70dd251e92901c43072",
GitTreeState:"dirty", BuildDate:"2017-08-29T20:32:58Z",
OpenPaasKubernetesVersion:"v1.03.02", GoVersion:"go1.7.5",
Compiler:"gc", Platform:"linux/amd64"}
```

docker info

如何获取有关环境和配置的各种信息？查看 kubectl cluster-info。

使用 docker 命令：

```
$ docker info
Containers: 40
Images: 168
Storage Driver: aufs
 Root Dir: /usr/local/google/docker/aufs
 Backing Filesystem: extfs
 Dirs: 248
 Dirperm1 Supported: false
Execution Driver: native-0.2
Logging Driver: json-file
Kernel Version: 3.13.0-53-generic
Operating System: Ubuntu 14.04.2 LTS
CPUs: 12
Total Memory: 31.32 GiB
Name: k8s-is-fun.mtv.corp.google.com
ID: ADUV:GCYR:B3VJ:HMPO:LNPQ:KD5S:YKFQ:76VN:IANZ:7TFV:ZBF4:BYJO
WARNING: No swap limit support
```

使用 kubectl 命令：

```
$ kubectl cluster-info
Kubernetes master is running at https://108.59.85.141
KubeDNS is running at
https://108.59.85.141/api/v1/namespaces/kube-
system/services/kube-dns/proxy
KubeUI is running at
https://108.59.85.141/api/v1/namespaces/kube-
system/services/kube-ui/proxy
```

Grafana is running at
https://108.59.85.141/api/v1/namespaces/kube-system/services/monitoring-grafana/proxy
Heapster is running at
https://108.59.85.141/api/v1/namespaces/kube-system/services/monitoring-heapster/proxy
InfluxDB is running at
https://108.59.85.141/api/v1/namespaces/kube-system/services/monitoring-influxdb/proxy