Reference Documentation

Design docs, concept definitions, and references for APIs and CLIs.

GUIDES REFERENCE SAMPLES SUPPORT

Search

kubectl for Docker Users

In this doc, we introduce the Kubernetes command line for interacting with the api to docker-cli users. The tool, kubectl, is designed to be familiar to docker-cli users but there are a few necessary differences. Each section of this doc highlights a docker subcommand explains the kubectl equivalent.

docker run

How do I run an nginx Deployment and expose it to the world? Checkout kubectl run.

With docker:

```
$ docker run -d --restart=always -e DOMAIN=cluster --name nginx-app -p 80:80 ngin›
a9ec34d9878748d2f33dc20cb25c714ff21da8d40558b45bfaec9955859075d0
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
a9ec34d98787        nginx               "nginx -g 'daemon of   2 seconds ago
```

```
$ kubectl run --image=nginx nginx-app --port=80 --env= DOMAIN=cluster
deployment "nginx-app" created
```

`kubectl run` creates a Deployment named "nginx" on Kubernetes cluster >= v1.2. If you are running older versions, it creates replication controllers instead. If you want to obtain the old behavior, use `--generator=run/v1` to create replication controllers. See `kubectl run` for more details. Note that `kubectl` commands will print the type and name of the resource created or mutated, which can then be used in subsequent commands. Now, we can expose a new Service with the deployment created above:

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

With kubectl, we create a Deployment which will make sure that N pods are running nginx (where N is the number of replicas stated in the spec, which defaults to 1). We also create a service with a selector that matches the Deployment's selector. See the Quick start for more information.

By default images are run in the background, similar to `docker run -d ...`, if you want to run things in the foreground, use:

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

Unlike `docker run ...`, if `--attach` is specified, we attach to `stdin`, `stdout` and `stderr`, there is no ability to control which streams are attached (`docker -a ...`).

Because we start a Deployment for your container, it will be restarted if you terminate the attached process (e.g. `ctrl-c`), this is different than `docker run -it`. To destroy the Deployment (and its pods) you need to run `kubectl delete deployment <name>`

docker ps

How do I list what is currently running? Checkout kubectl get.

With docker:

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
a9ec34d98787        nginx               "nginx -g 'daemon of     About an hour ago
```

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

docker attach

How do I attach to a process that is already running in a container? Checkout kubectl attach

With docker:

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
a9ec34d98787        nginx               "nginx -g 'daemon of   8 minutes ago
$ docker attach -it a9ec34d98787
...
```

With kubectl:

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

docker exec

How do I execute a command in a container? Checkout kubectl exec.

With docker:

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
a9ec34d98787        nginx               "nginx -g 'daemon of   8 minutes ago
$ docker exec a9ec34d98787 cat /etc/hostname
a9ec34d98787
```

With kubectl:

```
$ kubectl exec nginx-app-5jyvm -- cat /etc/hostname
nginx-app-5jyvm
```

What about interactive commands?

With docker:

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

With kubectl:

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

For more information see Getting into containers.

docker logs

How do I follow stdout/stderr of a running process? Checkout kubectl logs.

With docker:

```
$ docker logs -f a9e
192.168.9.1 - - [14/Jul/2015:01:04:02 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.
192.168.9.1 - - [14/Jul/2015:01:04:03 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.
```

With kubectl:

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

Now's a good time to mention slight difference between pods and containers; by default pods will not terminate if their processes exit. Instead it will restart the process. This is similar to the docker run option `--restart=always` with one major difference. In docker, the output for each invocation of the process is

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

See Logging for more information.

docker stop and docker rm

How do I stop and delete a running process? Checkout kubectl delete.

With docker

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
a9ec34d98787        nginx               "nginx -g 'daemon of   22 hours ago
$ docker stop a9ec34d98787
a9ec34d98787
$ docker rm a9ec34d98787
a9ec34d98787
```

With kubectl:

```
$ 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
```

Notice that we don't delete the pod directly. With kubectl we want to delete the Deployment that owns the pod. If we delete the pod directly, the Deployment will recreate the pod.

docker login

docker version

How do I get the version of my client and server? Checkout kubectl version.

With docker:

```
$ 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
```

With kubectl:

```
$ kubectl version
Client Version: version.Info{Major:"0", Minor:"20.1", GitVersion:"v0.20.1", GitCon
Server Version: version.Info{Major:"0", Minor:"21+", GitVersion:"v0.21.1-411-g326
```

docker info

How do I get miscellaneous info about my environment and configuration? Checkout kubectl cluster-info.

With docker:

```
 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
```

With kubectl:

```
$ kubectl cluster-info
Kubernetes master is running at https://108.59.85.141
KubeDNS is running at https://108.59.85.141/api/v1/proxy/namespaces/kube-system/se
KubeUI is running at https://108.59.85.141/api/v1/proxy/namespaces/kube-system/ser
Grafana is running at https://108.59.85.141/api/v1/proxy/namespaces/kube-system/se
Heapster is running at https://108.59.85.141/api/v1/proxy/namespaces/kube-system/s
InfluxDB is running at https://108.59.85.141/api/v1/proxy/namespaces/kube-system/s
```

I wish this page

enter your wish