Story of a `kubectl` command

Open Source Summit, Europe
22nd October, 2018
Hi, I'm Indra

Indradhanush Gupta
Software Engineer, Kinvolk

Github: indradhanush
Twitter: indradhanush92
Email: indra@kinvolk.io
Kinvolk

The Deep-stack Kubernetes Experts

Engineering services and products for Kubernetes, containers, process management and Linux user-space + kernel

Blog: kinvolk.io/blog
Github: kinvolk
Twitter: kinvolkio
Email: hello@kinvolk.io
Your take away from this talk?
Your take away from this talk?

1. What is Kubernetes?
2. What are the different components of Kubernetes?
3. What goes on behind the scenes of a kubectl command?
Your take away from this talk?

1. What is Kubernetes?
2. What are the different components of Kubernetes?
3. What goes on behind the scenes of a kubectl command?
Your take away from this talk?

1. What is Kubernetes?
2. What are the different components of Kubernetes?
3. What goes on behind the scenes of a kubectl command?
Your take away from this talk?

1. What is Kubernetes?
2. What are the different components of Kubernetes?
3. What goes on behind the scenes of a kubectl command?
What is Kubernetes?
Kubernetes

- Cluster manager
- Scheduler
- Orchestrator
Kubernetes

- Cluster manager
- Scheduler
- Orchestrator

...for containerized applications
A Kubernetes cluster
Kubernetes does not follow the UNIX philosophy
It does too many things!
And it can be overwhelming!
Container

App
Pod

App
Replicaset

App

App

App
Components in Master
Components in Master

API Server
Components in Master

- API Server
- Controller Manager
Components in Master

- API Server
- Controller Manager
- Scheduler
Components in Master

- etcd
- API Server
- Controller Manager
- Scheduler
Components in Worker

Node

Node

Node
Components in Worker

kubelet

Node

kubelet

Node

kubelet

Node
Components in Worker

- kube-proxy
Components in Worker

- kube-proxy
- kube-dns
```
kubectl create -f manifest.yaml
```
$ kubectl create deployment nginx --image=nginx
$ kubectl create deployment nginx --image=nginx
$ kubectl create deployment nginx --image=nginx
$ kubectl create deployment nginx --image=nginx
$ kubectl create deployment nginx --image=nginx

deployment.apps/nginx created
$ kubectl create deployment nginx --image=nginx

deployment.apps/nginx created

Imperative approach. Please don’t do this in production :)
```bash
$ kubectl get deployments

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESIRED</th>
<th>CURRENT</th>
<th>UP-TO-DATE</th>
<th>AVAILABLE</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>nginx</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0s</td>
</tr>
</tbody>
</table>
```
$ kubectl get deployments

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESIRED</th>
<th>CURRENT</th>
<th>UP-TO-DATE</th>
<th>AVAILABLE</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>nginx</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10s</td>
</tr>
</tbody>
</table>
$ kubectl get pods
```bash
$ kubectl get pods

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>nginx-65899c769f-58xbc</td>
<td>0/1</td>
<td>ContainerCreating</td>
<td>0</td>
<td>5s</td>
</tr>
</tbody>
</table>
```
$ kubectl get pods

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>nginx-65899c769f-58xbc</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>16s</td>
</tr>
</tbody>
</table>
kubectl
Client side validation

- Arguments
- Image name
- Manifest
Client side validation

- Arguments
- Image name
- Manifest (kubectl create -f)
And it’s time to send the request!
But where? 😐
API discovery

- OpenAPI schema
HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)

SITUATION:
THERE ARE 14 COMPETING STANDARDS.

14?! RIDICULOUS!
WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES. YEAH!

SITUATION:
THERE ARE 15 COMPETING STANDARDS.

https://xkcd.com/927/
API discovery

- OpenAPI schema

- https://www.openapis.org/about
API discovery

- Resources
- Group
- Version
Group: core
Group: core
Version: v1
$ kubectl get deployment nginx  -o yaml
$ kubectl get deployment nginx -o yaml
$ kubectl get deployment nginx -o yaml

apiVersion: extensions/v1beta1

kind: Deployment

....
$ kubectl get deployment nginx -o yaml

apiVersion: extensions/v1beta1

kind: Deployment

....
$ kubectl get deployment nginx -o yaml

apiVersion: extensions/v1beta1

kind: Deployment

....
$ kubectl get deployment nginx  -o yaml

apiVersion: extensions/v1beta1

kind: Deployment

....
Let’s take a verbose look at a request
$ kubectl get deployments -v 6
$ kubectl get deployments -v 6

[1021 08:53:04.617134 8299 loader.go:359] Config loaded from file /home/dhanush/.kube/config
$ kubectl get deployments -v 6

[2021-08-31T08:53:04.617134Z] Config loaded from file /home/dhanush/.kube/config

$ kubectl get deployments -v 6

[2021-08-04 08:53:04.617134 8299 loader.go:359] Config loaded from file /home/dhanush/.kube/config

$ kubectl get deployments -v 6

[08:53:04.617134] Config loaded from file /home/dhanush/.kube/config


$ kubectl get deployments -v 6

[2021-08-31 08:53:04.617134 8299 loader.go:359] Config loaded from file /home/dhanush/.kube/config


$ kubectl get deployments -v 6

[1021 08:53:04.617134 8299 loader.go:359] Config loaded from file /home/dhanush/.kube/config


$ kubectl get deployments -v 6

NAME    DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
nginx    1        1        1           1          2m
API discovery

- Cached at ~/.kube/cache
kubectl get pods -v 10
Client authentication

- Credentials from $KUBECONFIG
- Client certificates
- Bearer Tokens
- Username / Password
Server side authentication

- Client certificates
- Bearer Tokens
- Username / Password
Authorization chain

- Attribute Based Access Control
Authorization chain

- Attribute Based Access Control
- Role Based Access Control
Authorization chain

- Attribute Based Access Control
- Role Based Access Control
- Node
Authorization chain

- Attribute Based Access Control
- Role Based Access Control
- Node
- Webhook
Admission controllers

- Not a chain
Admission controllers

- Not a chain
- Modify or reject requests
Admission controllers

- Not a chain
- Modify or reject requests
- No role in read requests
Examples: Admission controllers

- AlwaysPullImages
- PodSecurityPolicy
kubectl → API Server → ✅
kubectl -> API Server -> etcd
kubectl ➔ API Server ➔ etcd

<namespace>/<name>
Initializers

- Dynamic controller
- Intercepts resource before creation
- Context specific logic
Initializers

$ kubectl get pods --include-uninitialized
Deployments controller

Deployment
Deployments controller

ReplicaSet
Deployment
Replicasets controller

ReplicaSet

Deployment
Replicasets controller

Pod
ReplicaSet
Deployment
Replicas = 3
State: Pending

NodeName: empty
Scheduler

- Filters pods with empty **NodeName**
Scheduler

- Filters pods with empty **NodeName**
- Filter worker nodes based on resources and affinity
Scheduler

- Filters pods with empty **nodeName**
- Filter worker nodes based on resources and affinity
- Prioritizes filtered worker nodes
Scheduler

- Filters pods with empty **NodeName**
- Filter worker nodes based on resources and affinity
- Prioritizes filtered worker nodes
- Choose node with highest priority
Scheduler

- Filters pods with empty NodeName
- Filter worker nodes based on resources and affinity
- Prioritizes filtered worker nodes
- Choose node with highest priority
- Creates Binding resource
Binding

NodeName
## Binding

<table>
<thead>
<tr>
<th>NodeName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Namespace</td>
</tr>
</tbody>
</table>
# Binding

<table>
<thead>
<tr>
<th>NodeName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Namespace</td>
</tr>
<tr>
<td>Pod Name &amp; UID</td>
</tr>
</tbody>
</table>

This table outlines the key components involved in the binding process.
kubelet → API Server
kubelet

Do you have a binding for me?

API Server
kubelet
kubelet

Pod
kubelet

Pod

Pause
Pause container (almost there!)

$ docker ps

<table>
<thead>
<tr>
<th>CONTAINER ID</th>
<th>IMAGE</th>
<th>COMMAND</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fccc6b7a99a</td>
<td>k8s.gcr.io/pause-amd64:3.1</td>
<td>&quot;/pause&quot;</td>
<td></td>
</tr>
</tbody>
</table>
Pause container

- Holds namespace for all containers of the pod
Pause container

- Holds namespace for all containers of the pod
- All application containers share the same namespaces
Pause container

- Holds namespace for all containers of the pod
- All application containers share the same namespaces
- Simplified intra pod networking
Pause container

- Holds namespace for all containers of the pod
- All application containers share the same namespaces
- Simplified intra pod networking
- Reap zombies if PID namespace sharing is enabled
Containers

- Pull the image
- Create the container
- Update Pod status
Summary

- Client side
  - Validation and Authentication
Summary

- Client side
  - Validation and Authentication

- Server side
  - Authentication
  - Authorization
Summary

- Admission controllers
Summary

- Admission controllers
- Write to etcd!
Summary

- Wait for Initializers 😴
Summary

- Wait for Initializers

- Deployments controller
  - Create ReplicaSet
Summary

- ReplicaSets controller
  - Create Pod
Summary

- Scheduler assigns a Node
Summary

- Scheduler assigns a Node

- Kubelet
  - Pause container
  - Application container
Thank you!

Indradhanush Gupta
Github: indradhanush
Twitter: indradhanush92
Email: indra@kinvolk.io

Kinvolk
Blog: kinvolk.io/blog
Github: kinvolk
Twitter: kinvolkio
Email: hello@kinvolk.io