KEYNOTE - API CITY 2018

TEACHING KUBERNETES API TO UNDERSTAND FASTIDIOUS SYSTEMS

ALENA HALL
@LENADROID

SEATTLE, WASHINGTON
OCTOBER 3 - 4, 2018


THE WORLD OF APIS
@LENADROID

APIS ARE IMPACTFUL TO THE WORLD

@LENADROID


"
THE MOST COMMON DECISION FACTOR I SEE IS "WHICH API I LIKE BETTER". APIS ARE HERE TO STAY AND YOU USE THEM 10 HOURS A DAY.
- GWEN SHAPIRA "
@LENADROID


USING APIS ENABLES US TO BE
EFFICIENT AT BUILDING GREAT THINGS
@LENADROID


BUILDING GREAT APIS EMPOWERS
OTHERS TO USE OUR SYSTEMS TO
BUILD GREAT THINGS
@LENADROID


WHAT ABOUT KUBERNETES
AND ITS API ?
@LENADROID

WHAT IS KUBERNETES?

WHAT SYSTEMS CAN RUN ON KUBERNETES?

IS IT THAT SIMPLE? *

* NO

PICKY

DISTRIBUTED SYSTEMS

@LENADROID

IF WE TEACH KUBERNETES – IT WILL LEARN!

WHAT HAPPENS
BEHIND THE SCENES?

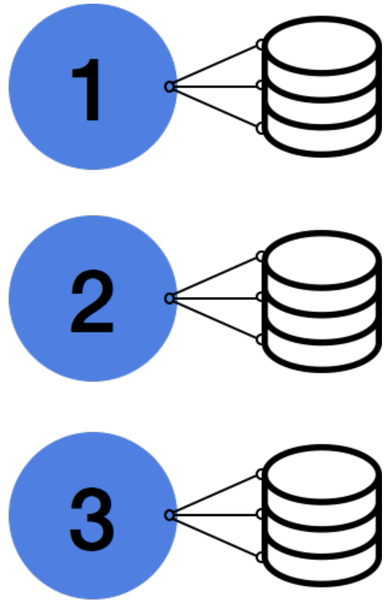
API OBJECTS

Pod

Job

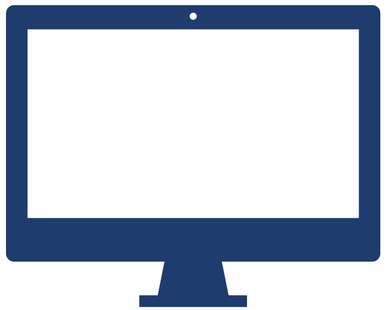
Deployment

Stateful Set

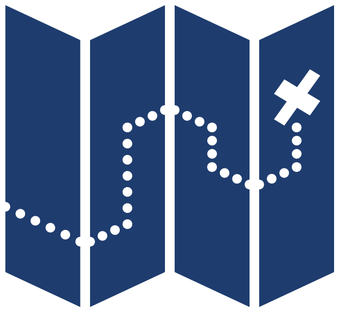


API OBJECTS

Service



ConfigMap



Secrets

Volumes

API OBJECTS

- Workloads APIs
- Service APIs
- Config and Storage APIs
- Metadata APIs
- Cluster APIs

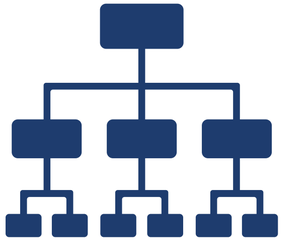
HOW DO WE RUN A SYSTEM ON KUBERNETES?

EXAMPLE: USING YAML FILES TO RUN A SYSTEM ON AZURE KUBERNETES SERVICE

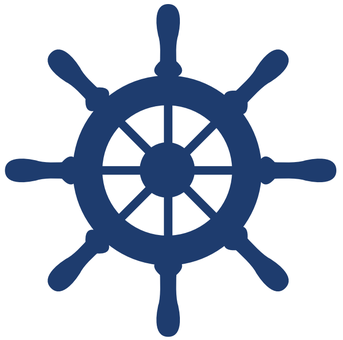
CONTROL PLANE
BEHIND THE SCENES

CONTROL PLANE

kube-api-server



kube-scheduler

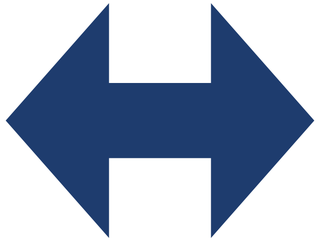


kube-controller manager

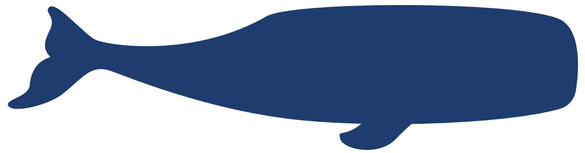
etcd

NODE COMPONENTS

kubelet



kube-proxy



Container Runtime

UP AND RUNNING

! =

OPERATING CORRECTLY

HOW DO WE FIX THIS?

ANSWER: EXTENSIBLE API

API EXTENSIBILITY: WHAT'S POSSIBLE?

API Extensions:

- User-Defined Types
- Combining New APIs with Automation
- Changing Built-in Resources
- API Access Extensions
- Authentication
- Authorization
- Dynamic Admission Control

Infrastructure Extensions:

- Storage Plugins
- Device Plugins
- Network Plugins
- Scheduler Extensions

CUSTOM RESOURCE DEFINITIONS

(CRDs)

CUSTOM CONTROLLERS

OPERATOR

CRD + CUSTOM CONTROLLER

EXAMPLE

RUNNING A TENSORFLOW JOB WITH KUBEFLOW AND TF-JOB OPERATOR ON AZURE KUBERNETES SERVICE

Kubeflow quick start

KUBEFLOW.ORG/DOCS/STARTED/GETTING-STARTED

Requirements:

- ksonnet version 0.11.0 or later.
- Kubernetes 1.8 or later
- kubectl

Download, set up, and deploy:

1. Run the following script to download `kfctl.sh`:

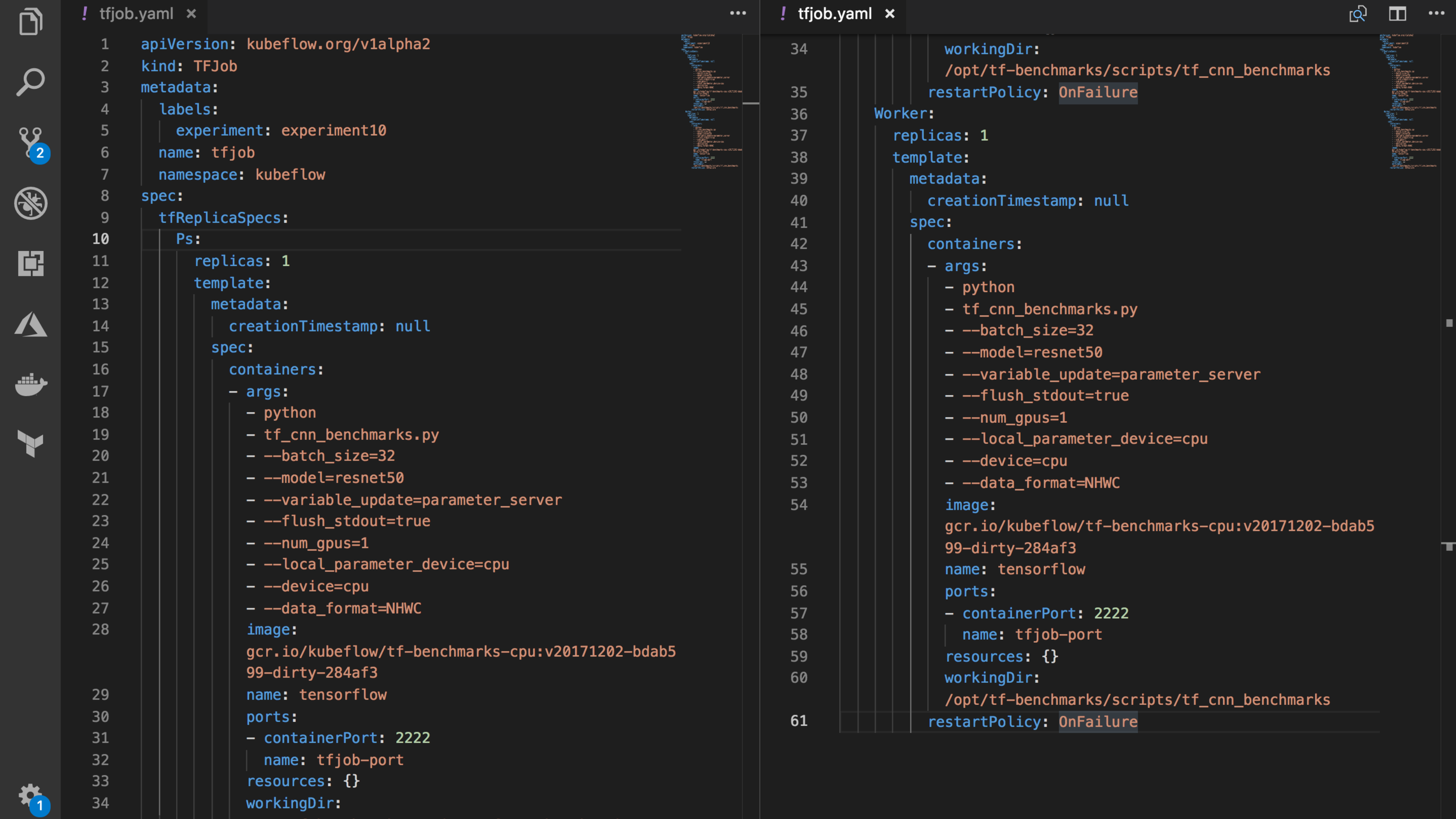
```
mkdir ${KUBEFLOW_SRC}
cd ${KUBEFLOW_SRC}
export KUBEFLOW_TAG=<version>
curl https://raw.githubusercontent.com/kubeflow/kubeflow/${KUBEFLOW_TAG}/scripts/download.sh | bash
```

   - **KUBEFLOW_SRC** a directory where you want to download the source to
   - **KUBEFLOW_TAG** a tag corresponding to the version to check out, such as `master` for the latest code.
   - **Note** you can also just clone the repository using git.

2. Run the following scripts to set up and deploy Kubeflow:

```
${KUBEFLOW_REPO}/scripts/kfctl.sh init ${KFAPP} --platform none
cd ${KFAPP}
${KUBEFLOW_REPO}/scripts/kfctl.sh generate k8s
${KUBEFLOW_REPO}/scripts/kfctl.sh apply k8s
```

   - **${KFAPP}** The name of a directory to store your configs. This directory will be created when you run init.
     - The ksonnet app will be created in the directory **${KFAPP}/ks_app**


tfjob.yaml
apiVersion: kubeflow.org/v1alpha2
kind: TFJob
metadata:
labels:
experiment: experiment10
name: tfjob
namespace: kubeflow
spec:
tfReplicaSpecs:
Ps:
replicas: 1
template:
metadata:
creationTimestamp: null
spec:
containers:
- args:
- python
- tf_cnn_benchmarks.py
- --batch_size=32
- --model=resnet50
- --variable_update=parameter_server
- --flush_stdout=true
- --num_gpus=1
- --local_parameter_device=cpu
- --device=cpu
- --data_format=NHWC
image:
gcr.io/kubeflow/tf-benchmarks-cpu:v20171202-bdab5
99-dirty-284af3
name: tensorflow
ports:
- containerPort: 2222
name: tfjob-port
resources: {}
workingDir:
/opt/tf-benchmarks/scripts/tf_cnn_benchmarks
restartPolicy: OnFailure
Worker:
replicas: 1
template:
metadata:
creationTimestamp: null
spec:
containers:
- args:
- python
- tf_cnn_benchmarks.py
- --batch_size=32
- --model=resnet50
- --variable_update=parameter_server
- --flush_stdout=true
- --num_gpus=1
- --local_parameter_device=cpu
- --device=cpu
- --data_format=NHWC
image:
gcr.io/kubeflow/tf-benchmarks-cpu:v20171202-bdab5
99-dirty-284af3
name: tensorflow
ports:
- containerPort: 2222
name: tfjob-port
resources: {}
workingDir:
/opt/tf-benchmarks/scripts/tf_cnn_benchmarks
restartPolicy: OnFailure

BEHIND THE SCENES

TF-OPERATOR

CRD

```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: tfjobs.kubeflow.org
spec:
  group: kubeflow.org
  version: v1alpha1
  names:
    kind: TFJob
    singular: tfjob
    plural: tfjobs
```

GITHUB.COM/KUBEFLOW/TF-OPERATOR

TFJOB

```
// TFJob represents the configuration of signal TFJob
type TFJob struct {
        metav1.TypeMeta `json:",inline"`

        // Standard object's metadata.
        metav1.ObjectMeta `json:"metadata,omitempty"`

        // Specification of the desired behavior of the TFJob.
        Spec TFJobSpec `json:"spec,omitempty"`

        // Most recently observed status of the TFJob.
        // This data may not be up to date.
        // Populated by the system.
        // Read-only.
        Status TFJobStatus `json:"status,omitempty"`
}
```

GITHUB.COM/KUBEFLOW/TF-OPERATOR

```
// TFJobSpec is a desired state description of the TFJob.
type TFJobSpec struct {
        // TFReplicaSpecs is map of TFReplicaType and TFReplicaSpec
        // specifies the TF replicas to run.
        // For example,
        //   {
        //     "PS": TFReplicaSpec,
        //     "Worker": TFReplicaSpec,
        //   }
        TFReplicaSpecs map[TFReplicaType]*TFReplicaSpec `json:"tfReplicaSpecs"`
}
```

GITHUB.COM/KUBEFLOW/TF-OPERATOR

```
// TFReplicaType is the type for TFReplica.
type TFReplicaType string

const (
        // TFReplicaTypePS is the type for parameter servers of distributed TensorFlow.
        TFReplicaTypePS TFReplicaType = "PS"

        // TFReplicaTypeWorker is the type for workers of distributed TensorFlow.
        // This is also used for non-distributed TensorFlow.
        TFReplicaTypeWorker TFReplicaType = "Worker"

        // TFReplicaTypeChief is the type for chief worker of distributed TensorFlow.
        // If there is "chief" replica type, it's the "chief worker".
        // Else, worker:0 is the chief worker.
        TFReplicaTypeChief TFReplicaType = "Chief"

        // TFReplicaTypeEval is the type for evaluation replica in TensorFlow.
        TFReplicaTypeEval TFReplicaType = "Eval"
)
```

GITHUB.COM/KUBEFLOW/TF-OPERATOR

```
// RestartPolicy describes how the TFReplicas should be restarted.
// Only one of the following restart policies may be specified.
// If none of the following policies is specified, the default one
// is RestartPolicyAlways.
type RestartPolicy string

const (
        RestartPolicyAlways    RestartPolicy = "Always"
        RestartPolicyOnFailure RestartPolicy = "OnFailure"
        RestartPolicyNever     RestartPolicy = "Never"

        // `ExitCode` policy means that user should add exit code by themselves,
        // `tf-operator` will check these exit codes to
        // determine the behavior when an error occurs:
        // - 1-127: permanent error, do not restart.
        // - 128-255: retryable error, will restart the pod.
        RestartPolicyExitCode RestartPolicy = "ExitCode"
)
```

GITHUB.COM/KUBEFLOW/TF-OPERATOR

APIS ARE IMPACTFUL

- Hightly flexible and customizeable system
- Multiple extension points and approaches
- Transparent API favor extensibility
- Convenient to add new and missing functionality

THANK YOU!

My blog - lenadroid.github.io/posts.html

My talk on Stateful Sets, Persistent Volumes, Persistent Volime Claims, Storage Classes - aka.ms/gotochgo

Thomas Stringer's post on Custom Controllers – aka.ms/custom-controllers

ALENA HALL - LENADROID

- ✓ Works on Azure at Microsoft
- ✓ Lives in Seattle
- ✓ F# Software Foundation Board of Trustees
- ✓ Organizes @ML4ALL
- ✓ Has a channel: YouTube/c/AlenaHall