# **How to Serve Machine Learning Models with TensorFlow Serving and Docker**

# **Welcome!**

#### Deploying Tensorflow model with kubeflow using TF serving

Difficulty: Intermediate Estimated Time: 10 minutes

In this scenario, you will learn how to serve a Machine Learning Model with TensorFlow (TF) Serving using Kubeflow. TF Serving is the recommended way to serve TensorFlow models. TF Serving enables users to quickly deploy models to production environments. We will use the interactive environment to create a single-node Kubernetes cluster allowing you to experience Kubeflow and understand how to deploy models on a model server with gRPC/REST endpoints.

The goal of this scenario is to understand how to serve a model using TF Serving on Kubeflow.

<https://github.com/juju-solutions/charm-tf-serving>

<https://github.com/juju-solutions/bundle-kubeflow>

# **Step 1: Setup**

Before getting started, first install kubernetes, docker and tensorflow model server.

**Install kubernetes**

we will use microk8s (a Low-ops, minimal production Kubernetes), for our scenario

### **Install docker**

$ sudo apt-get update

$ sudo apt-get install \

apt-transport-https \

ca-certificates \

curl \

gnupg-agent \

software-properties-common

$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -

$ sudo apt-key fingerprint 0EBFCD88

$ sudo add-apt-repository \

"deb [arch=amd64] https://download.docker.com/linux/debian \

$(lsb\_release -cs) \

stable"

$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli containerd.io

Verify that Docker Engine is installed correctly by running the hello-world image.

$ sudo docker run hello-world

### **Install TensorFlow Serving**

This is all you need - one command line!

apt-get install tensorflow-model-server

**Step 2: Prepare servable model:**

For this scenario we will deploy a pretrained saved model from Tensorflow/models. We will use a toy model called Half plus two which generates 0.5\*x+2 for the values of x we provide for prediction:

To get this model, first clone the TensorFlow Serving repo.

mkdir -p /tmp/tfserving

cd /tmp/tfserving

git clone <https://github.com/tensorflow/serving>

Set path to the model.

source=/tmp/tfserving/serving/tensorflow\_serving/servables/tensorflow/testdata/saved\_model\_half\_plus\_two\_cpu,\

target=/models/half\_plus\_two \

**Step 3: create Kubernetes resources and deploy model**  
To deploy a model we create following resources

* A deployment to deploy the model using TFServing
* A K8s service to create an endpoint a service
* An Istio virtual service to route traffic to the model and expose it through the Istio gateway
* An Istio DestinationRule is for doing traffic splitting.

Clone this manifests to create resources for our deployment and serve the model.

$ git clone --recursive <https://github.com/twarik/maven.git>

curl -s

[https://github.com/twarik/maven](https://github.com/twarik/maven.git)/resources.yaml

Microk8s kubectl create -f resources.yaml

### Step 4: Sending prediction request directly

### Make REST requests

If the service type is LoadBalancer, it will have its own accessible external ip. Get the external ip by:

kubectl get svc mnist-service

And then send the request

curl -X POST -d @input.json <http://EXTERNAL_IP:8500/v1/models/mnist:predic>t

Details of the project can be found at https://github.com/kubeflow/kubeflow

# **Creating a servable model**

To start, clone this git repository locally <https://github.com/tensorflow/serving>:

git clone https://github.com/tensorflow/serving.git

It has example models that we'll be deploying down below.

For this scenario we will deploy a pretrained ResNet saved model from Tensorflow/models.

*# Download the ResNet saved models* from the Google Storage then extract it:

**mkdir /tmp/myresnet**  
curl -s <https://storage.googleapis.com/download.tensorflow.org/models/official/20181001_resnet/savedmodels/resnet_v2_fp32_savedmodel_NHWC_jpg.tar.gz> | tar --strip-components=2 -C /tmp/myresnet -xvz

tar xfz inception-v3–2016–03–01.tar.gz  
$ cd -

## **Serving a model** To deploy a model we create following resources as illustrated below

* A deployment to deploy the model using TFServing
* A K8s service to create an endpoint a service
* An Istio virtual service to route traffic to the model and expose it through the Istio gateway
* An Istio DestinationRule is for doing traffic splitting.

t

### Create Kubernetes Deployment and Service

$ git clone --recursive <https://github.com/tensorflow/tensorflow.git>

In the resources folder, you will find 4 manifests to create resources for our deployment.

Microk8s kubectl create -f resources/deployment.yaml  
Microk8s kubectl create -f resources/Service.yaml  
Microk8s kubectl create -f resources/DestinationRule.yaml  
Microk8s kubectl create -f resources/VirtualService.yaml

kubectl apply -f tf\_deployment.yml

### Sending prediction request directly

If the service type is LoadBalancer, it will have its own accessible external ip. Get the external ip by:

kubectl get svc mnist-service

And then send the request

curl -X POST -d @input.json http://EXTERNAL\_IP:8500/v1/models/mnist:predict

To start, clone this git repository locally <https://github.com/tensorflow/serving>:

git clone https://github.com/tensorflow/serving.git

It has example models that we'll be deploying down below.