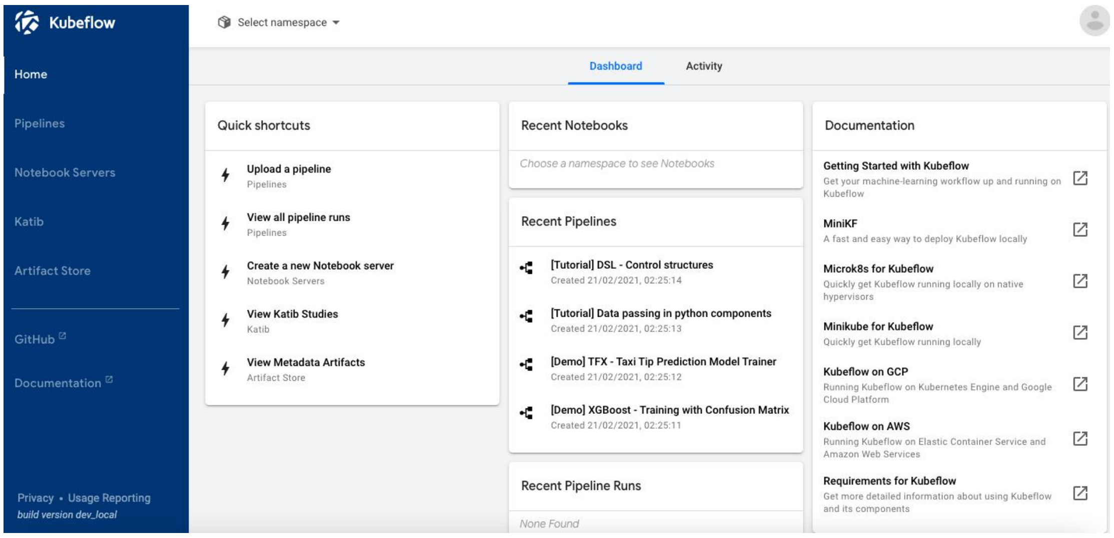
Kubernets   - > KubeFlow For Machine Learning


Kubeflow

Select namespace

Home

Pipelines

Notebook Servers

Katib

Artifact Store

GitHub

Documentation

Privacy • Usage Reporting
build version dev_local

Dashboard   Activity

Quick shortcuts

- Upload a pipeline — Pipelines
- View all pipeline runs — Pipelines
- Create a new Notebook server — Notebook Servers
- View Katib Studies — Katib
- View Metadata Artifacts — Artifact Store

Recent Notebooks

Choose a namespace to see Notebooks

Recent Pipelines

- [Tutorial] DSL - Control structures — Created 21/02/2021, 02:25:14
- [Tutorial] Data passing in python components — Created 21/02/2021, 02:25:13
- [Demo] TFX - Taxi Tip Prediction Model Trainer — Created 21/02/2021, 02:25:12
- [Demo] XGBoost - Training with Confusion Matrix — Created 21/02/2021, 02:25:11

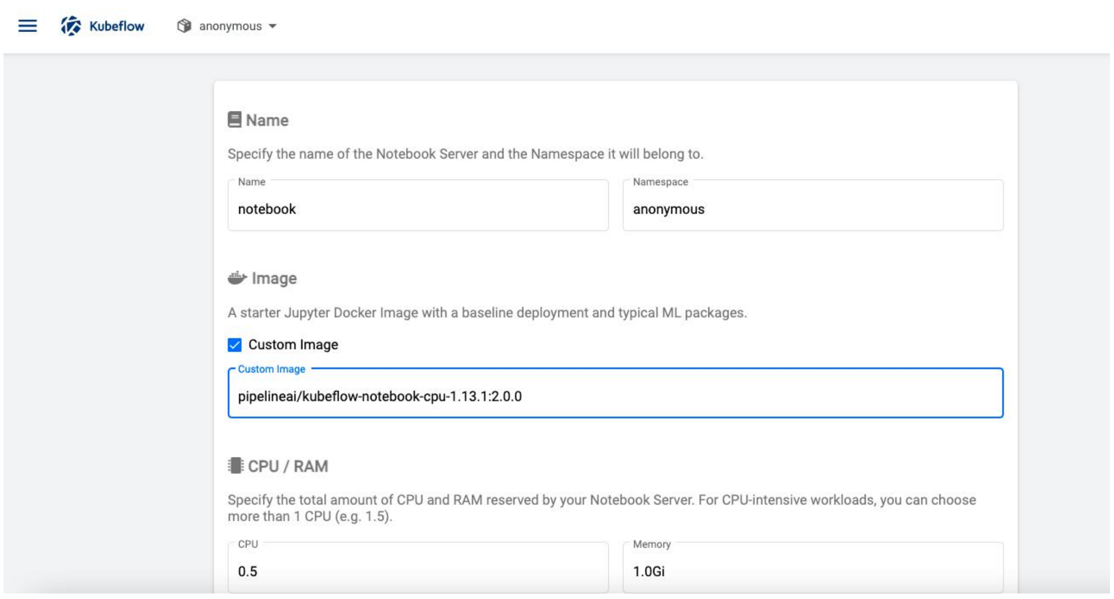
Recent Pipeline Runs

None Found

Documentation

- Getting Started with Kubeflow — Get your machine-learning workflow up and running on Kubeflow
- MiniKF — A fast and easy way to deploy Kubeflow locally
- Microk8s for Kubeflow — Quickly get Kubeflow running locally on native hypervisors
- Minikube for Kubeflow — Quickly get Kubeflow running locally
- Kubeflow on GCP — Running Kubeflow on Kubernetes Engine and Google Cloud Platform
- Kubeflow on AWS — Running Kubeflow on Elastic Container Service and Amazon Web Services
- Requirements for Kubeflow — Get more detailed information about using Kubeflow and its components

Select the Create Notebook Server --->


Kubeflow   anonymous

Name

Specify the name of the Notebook Server and the Namespace it will belong to.

Name: notebook

Namespace: anonymous

Image

A starter Jupyter Docker Image with a baseline deployment and typical ML packages.

☑ Custom Image

Custom Image: pipelineai/kubeflow-notebook-cpu-1.13.1:2.0.0

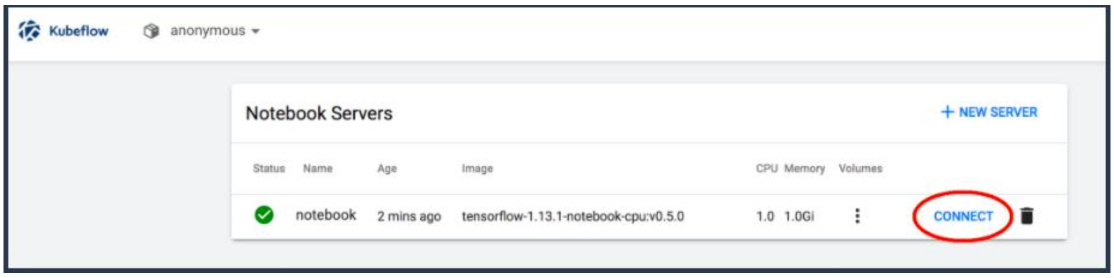
CPU / RAM

Specify the total amount of CPU and RAM reserved by your Notebook Server. For CPU-intensive workloads, you can choose more than 1 CPU (e.g. 1.5).

CPU: 0.5

Memory: 1.0Gi

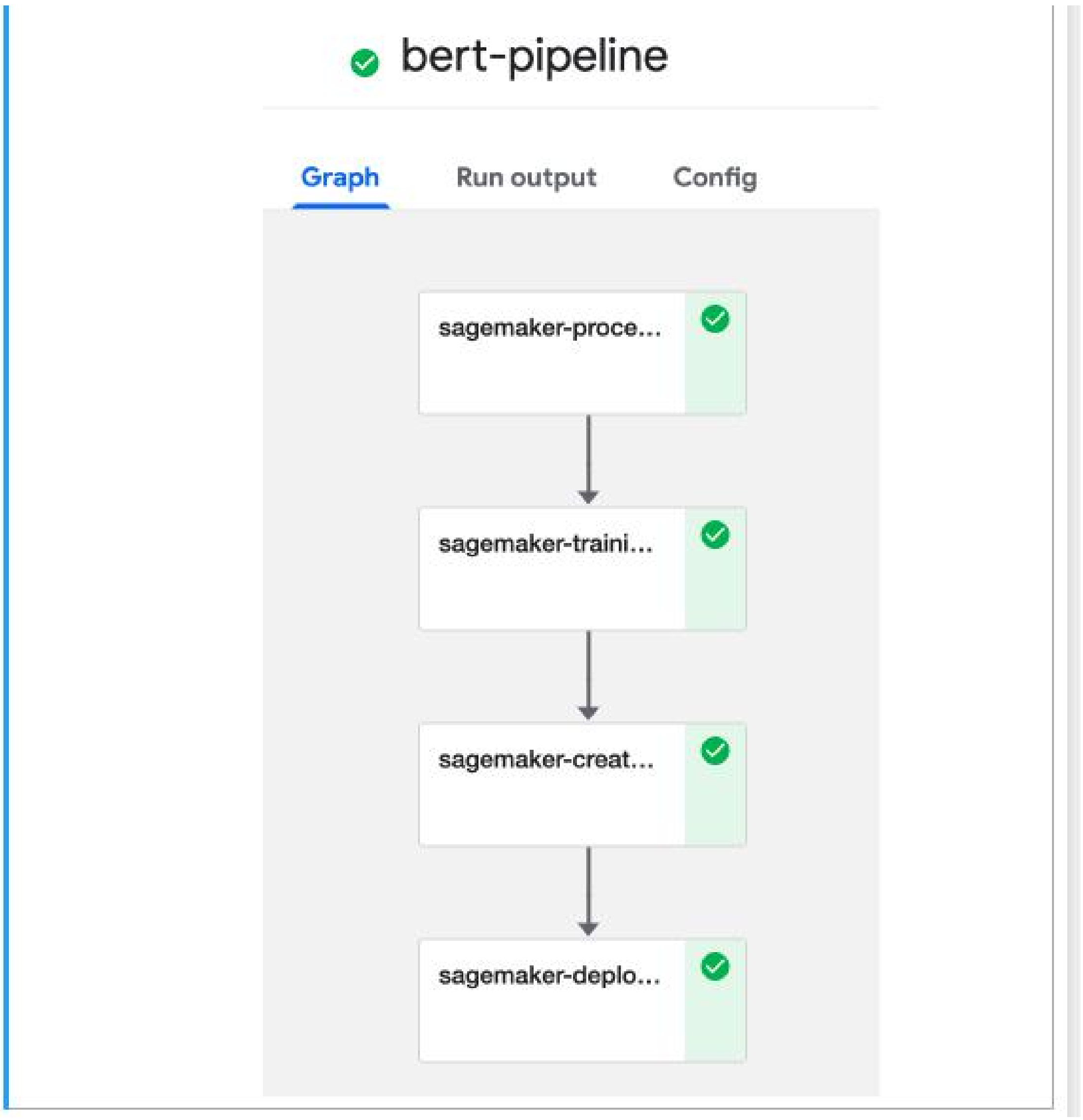
Choose CPU as 1.0 and Press Enter



Notebook Servers

Status	Name	Age	Image	CPU	Memory	Volumes
✓	notebook	2 mins ago	tensorflow-1.13.1-notebook-cpu:v0.5.0	1.0	1.0Gi	⋮

We need to connect and this will launch to Jupyter Notebooks



Build a Simple Pipeline

In this example, we want to calculate sum of three numbers

Let's assume we have a python image to use. It accepts two arguments and return sum of them

The sum of a and b will be used to calculate final result with sum of c and d. In total, we will have three arithmetical operators. Then we use another echo operator to print the result.

Create a Container Image for Each Component

Assumes that you have already created a program to perform the task required in a particular step of your ML workflow. For example, if the task is to train an ML model, then you must have a program that does the training,

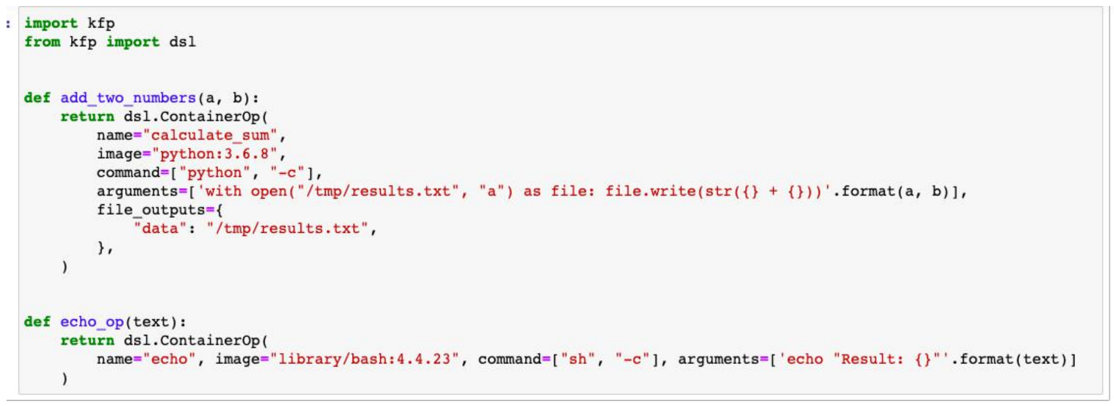
Your component can create outputs that the downstream components can use as inputs.

This will be used to build Job Directed Acyclic Graph (DAG)

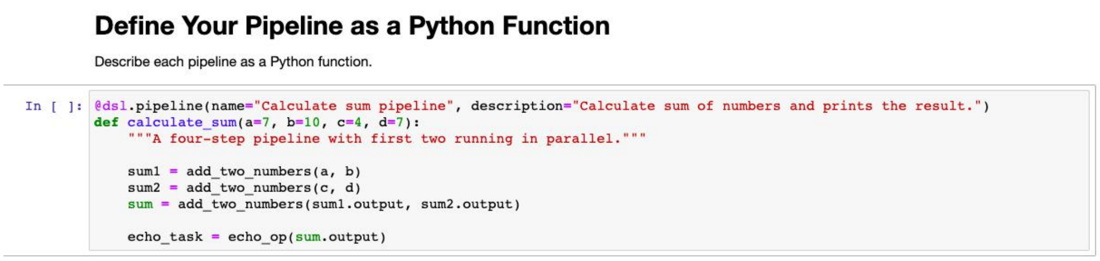
Create a Python Function to Wrap Your Component

Define a Python function to describe the interactions with the Docker container image that contains your pipeline component.

Here, in order to simplify the process, we use simple way to calculate sum. Ideally, you need to build a new container image for your code change.



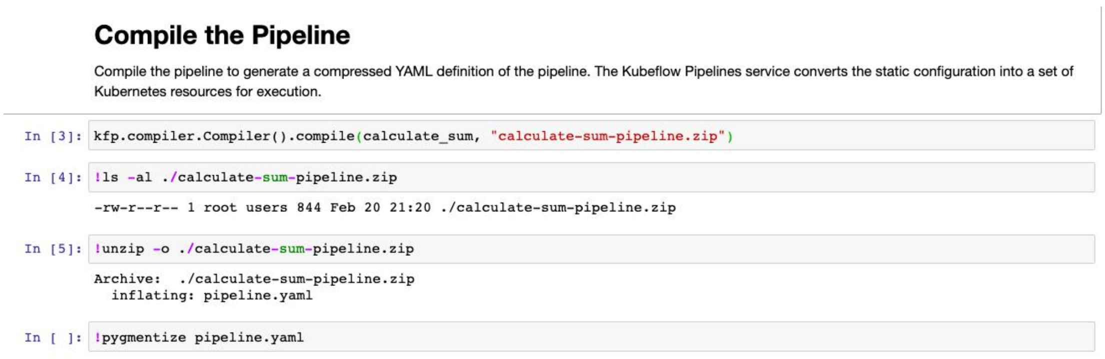
```
import kfp
from kfp import dsl


def add_two_numbers(a, b):
    return dsl.ContainerOp(
        name="calculate_sum",
        image="python:3.6.8",
        command=["python", "-c"],
        arguments=['with open("/tmp/results.txt", "a") as file: file.write(str({} + {}))'.format(a, b)],
        file_outputs={
            "data": "/tmp/results.txt",
        },
    )


def echo_op(text):
    return dsl.ContainerOp(
        name="echo", image="library/bash:4.4.23", command=["sh", "-c"], arguments=['echo "Result: {}"'.format(text)]
    )
```



Define Your Pipeline as a Python Function

Describe each pipeline as a Python function.

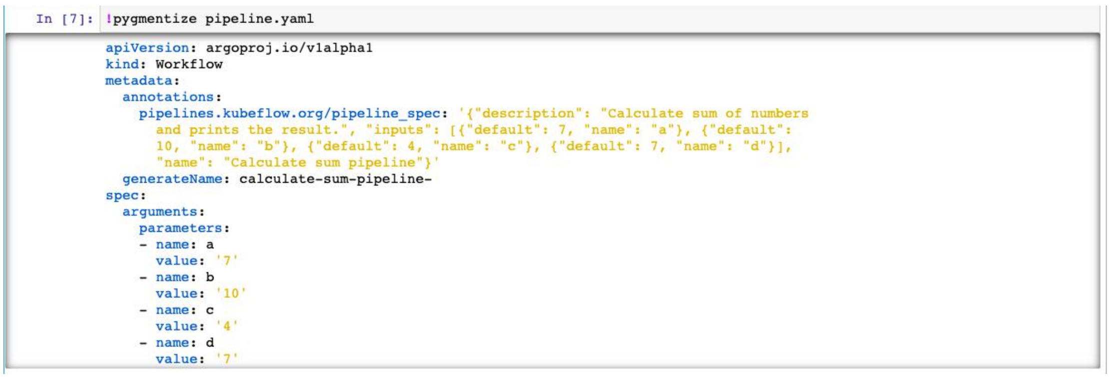
```
In [ ]: @dsl.pipeline(name="Calculate sum pipeline", description="Calculate sum of numbers and prints the result.")
        def calculate_sum(a=7, b=10, c=4, d=7):
            """A four-step pipeline with first two running in parallel."""

            sum1 = add_two_numbers(a, b)
            sum2 = add_two_numbers(c, d)
            sum = add_two_numbers(sum1.output, sum2.output)

            echo_task = echo_op(sum.output)
```



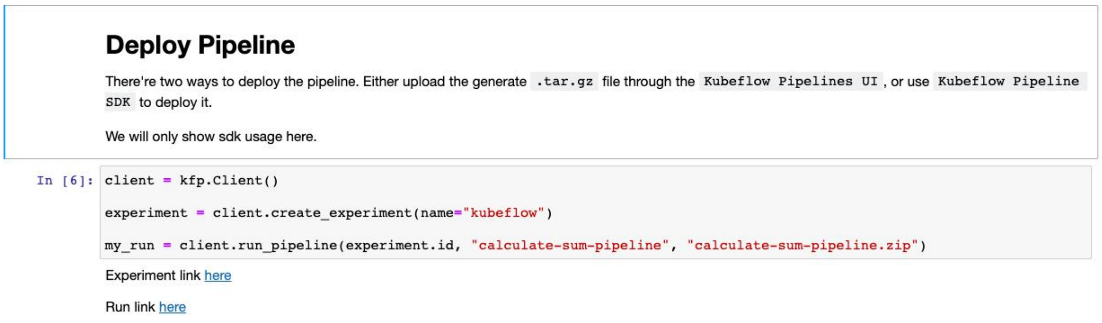
Compile the Pipeline

Compile the pipeline to generate a compressed YAML definition of the pipeline. The Kubeflow Pipelines service converts the static configuration into a set of Kubernetes resources for execution.

```
In [3]: kfp.compiler.Compiler().compile(calculate_sum, "calculate-sum-pipeline.zip")

In [4]: !ls -al ./calculate-sum-pipeline.zip

        -rw-r--r-- 1 root users 844 Feb 20 21:20 ./calculate-sum-pipeline.zip

In [5]: !unzip -o ./calculate-sum-pipeline.zip

        Archive:  ./calculate-sum-pipeline.zip
          inflating: pipeline.yaml

In [ ]: !pygmentize pipeline.yaml
```



```
In [7]: !pygmentize pipeline.yaml

        apiVersion: argoproj.io/v1alpha1
        kind: Workflow
        metadata:
          annotations:
            pipelines.kubeflow.org/pipeline_spec: '{"description": "Calculate sum of numbers
              and prints the result.", "inputs": [{"default": 7, "name": "a"}, {"default":
              10, "name": "b"}, {"default": 4, "name": "c"}, {"default": 7, "name": "d"}],
              "name": "Calculate sum pipeline"}'
          generateName: calculate-sum-pipeline-
        spec:
          arguments:
            parameters:
            - name: a
              value: '7'
            - name: b
              value: '10'
            - name: c
              value: '4'
            - name: d
              value: '7'
```



Deploy Pipeline

There're two ways to deploy the pipeline. Either upload the generate `.tar.gz` file through the `Kubeflow Pipelines UI`, or use `Kubeflow Pipeline SDK` to deploy it.

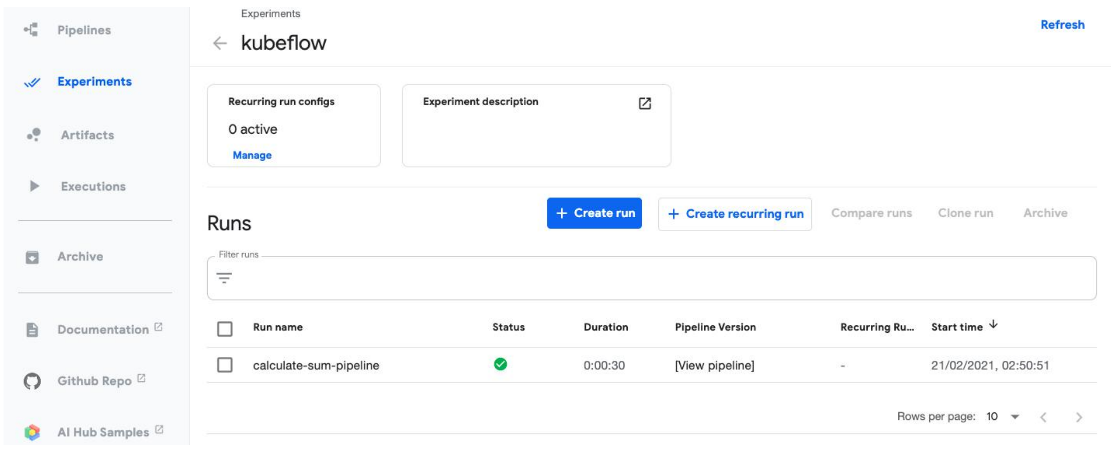
We will only show sdk usage here.

```
In [6]: client = kfp.Client()

        experiment = client.create_experiment(name="kubeflow")

        my_run = client.run_pipeline(experiment.id, "calculate-sum-pipeline", "calculate-sum-pipeline.zip")
```

Experiment link here

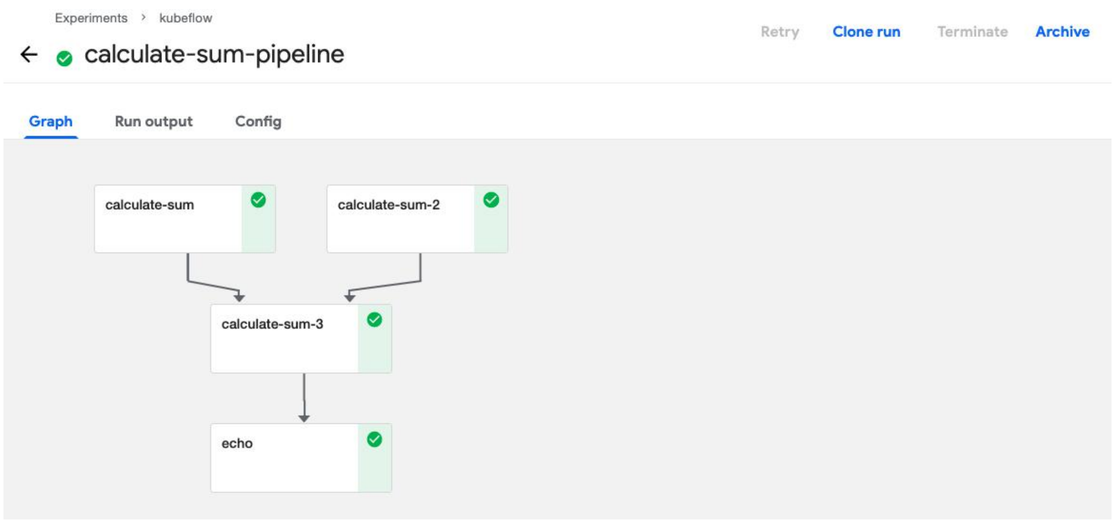
Run link here

Experiment link Shown Below:

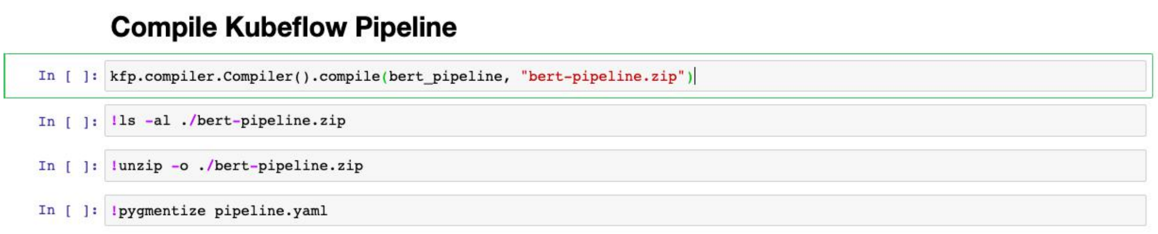


Run name	Status	Duration	Pipeline Version	Recurring Ru...	Start time
calculate-sum-pipeline	✓	0:00:30	[View pipeline]	-	21/02/2021, 02:50:51

Run Link Shown Below:



Create a Kubeflow Pipeline with BERT and Amazon SageMaker



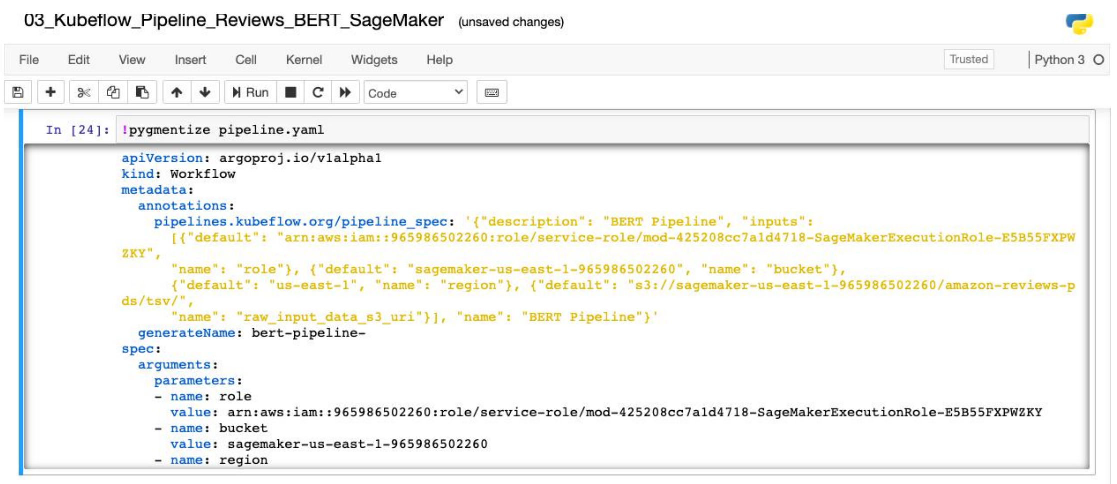
Compile Kubeflow Pipeline

```
In [ ]: kfp.compiler.Compiler().compile(bert_pipeline, "bert-pipeline.zip")
```

```
In [ ]: !ls -al ./bert-pipeline.zip
```

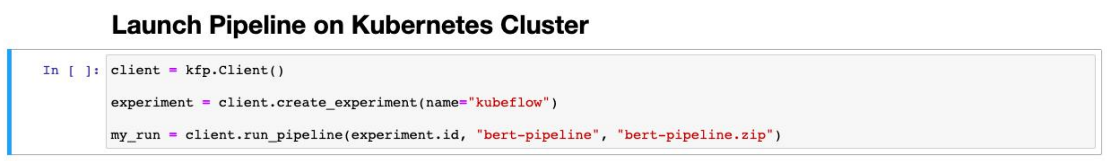
```
In [ ]: !unzip -o ./bert-pipeline.zip
```

```
In [ ]: !pygmentize pipeline.yaml
```


03_Kubeflow_Pipeline_Reviews_BERT_SageMaker (unsaved changes)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

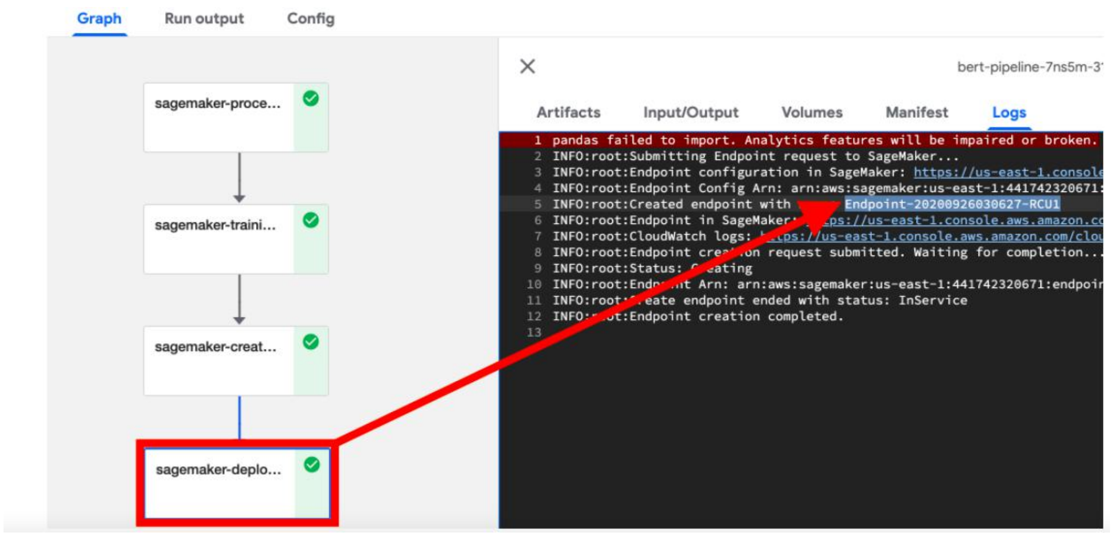
```
In [24]: !pygmentize pipeline.yaml
```

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  annotations:
    pipelines.kubeflow.org/pipeline_spec: '{"description": "BERT Pipeline", "inputs":
      [{"default": "arn:aws:iam::965986502260:role/service-role/mod-425208cc7a1d4718-SageMakerExecutionRole-E5B55FXPW
ZKY",
      "name": "role"}, {"default": "sagemaker-us-east-1-965986502260", "name": "bucket"},
      {"default": "us-east-1", "name": "region"}, {"default": "s3://sagemaker-us-east-1-965986502260/amazon-reviews-p
ds/tsv/",
      "name": "raw_input_data_s3_uri"}], "name": "BERT Pipeline"}'
  generateName: bert-pipeline-
spec:
  arguments:
    parameters:
    - name: role
      value: arn:aws:iam::965986502260:role/service-role/mod-425208cc7a1d4718-SageMakerExecutionRole-E5B55FXPWZKY
    - name: bucket
      value: sagemaker-us-east-1-965986502260
    - name: region
```



Launch Pipeline on Kubernetes Cluster

```
In [ ]: client = kfp.Client()

        experiment = client.create_experiment(name="kubeflow")

        my_run = client.run_pipeline(experiment.id, "bert-pipeline", "bert-pipeline.zip")
```



```
pandas failed to import. Analytics features will be impaired or broken.
INFO:root:Submitting Endpoint request to SageMaker...
INFO:root:Endpoint configuration in SageMaker: https://us-east-1.console
INFO:root:Endpoint Config Arn: arn:aws:sagemaker:us-east-1:441742320671:
INFO:root:Created endpoint with      Endpoint-20200926030627-RCU1
INFO:root:Endpoint in SageMaker: https://us-east-1.console.aws.amazon.co
INFO:root:CloudWatch logs: https://us-east-1.console.aws.amazon.com/clou
INFO:root:Endpoint creation request submitted. Waiting for completion...
INFO:root:Status: Creating
INFO:root:Endpoint Arn: arn:aws:sagemaker:us-east-1:441742320671:endpoi
INFO:root:Create endpoint ended with status: InService
INFO:root:Endpoint creation completed.
```

Finally We can Predict the results:

```
In [26]: import boto3

         sm_runtime = boto3.Session(region_name=region).client("sagemaker-runtime")

         endpoint_name = "<COPY-AND-PASTE-FROM-KUBEFLOW-PIPELINE-LOGS>"
```

```
In [27]: import json

         inputs = [{"features": ["This is great!"]}, {"features": ["This is bad."]}]

         response = sm_runtime.invoke_endpoint(
             EndpointName=endpoint_name,
             ContentType="application/jsonlines",
             Accept="application/jsonlines",
             Body=json.dumps(inputs).encode("utf-8"),
         )
         print("response: {}".format(response))

         predicted_classes_str = response["Body"].read().decode()
         predicted_classes_json = json.loads(predicted_classes_str)

         predicted_classes = predicted_classes_json.splitlines()
         print("predicted_classes: {}".format(predicted_classes))

         for predicted_class_json, input_data in zip(predicted_classes, inputs):
             predicted_class = json.loads(predicted_class_json)["predicted_label"]
             print('Predicted star_rating: {} for review_body "{}"'.format(predicted_class, input_data["features"][0]))
```