Running ML Pipelines on Kubeflow 2.5

Objectives

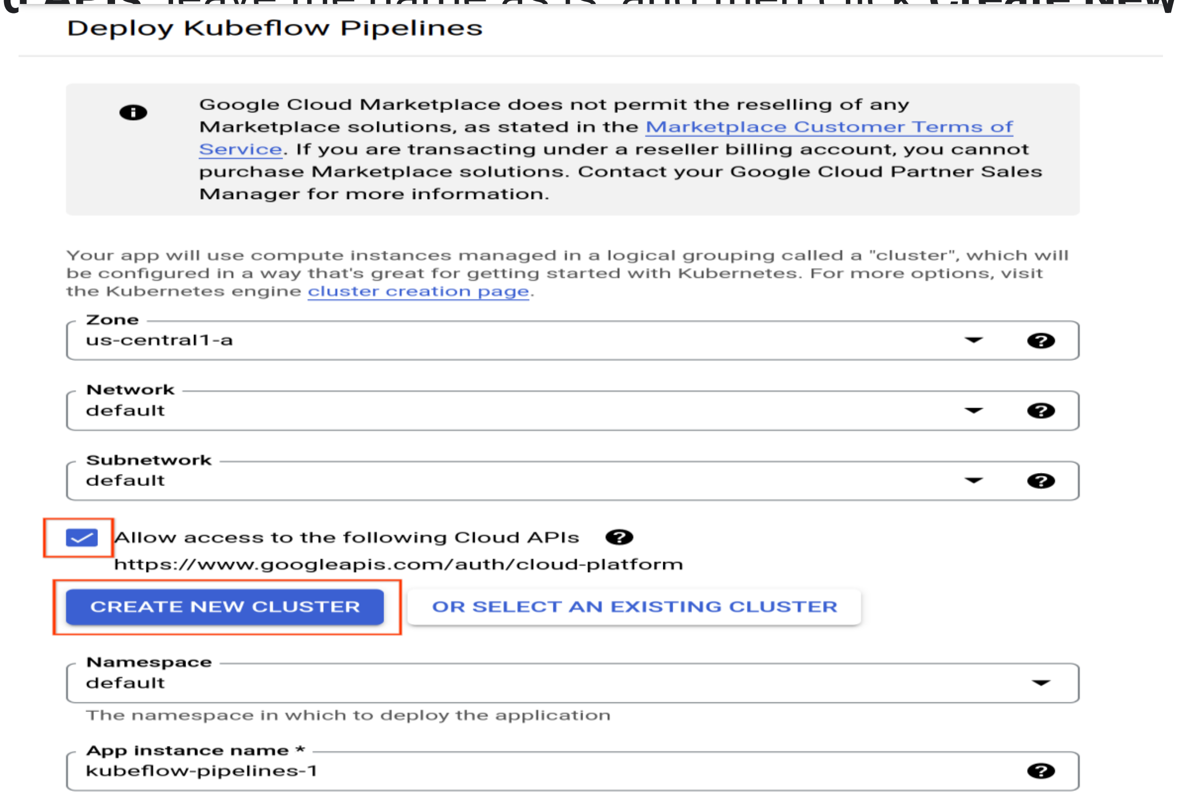
In this lab, you perform the following tasks:

- Create a Kubernetes cluster and configure AI Platform pipelines
- Launch pipelines dashboard
- Create and run an experiment from an example end-to-end ML Pipeline
- Examine and verify the output of each step
- Inspect the pipeline graph, various metrics, logs, charts and parameters

Task 1. Set up an AI Platform Pipelines instance

In this task, you deploy Kubeflow Pipelines as a Kuberenetes App, which are solutions with simple click to deploy to Google Kubernetes Engine and that have the flexibility to deploy to Kubernetes clusters on-premises or in third-party clouds. You will see Kubeflow Pipelines integrated into your Google Cloud environment as **AI Platform Pipelines**. If interested, learn more about Kubeflow Pipelines in the documentation during installation steps.

1.From the the Navigation menu, scroll down to **AI Platform** and pin the section for easier access later in the lab.
2.Click **Pipelines**.
3.Then click **New Instance**.
4.Click **Configure**.
5.Check **Allow access to the following Cloud APIs**, leave the name as is, and then click **Create New Cluster**.



Deploy Kubeflow Pipelines

Google Cloud Marketplace does not permit the reselling of any Marketplace solutions, as stated in the Marketplace Customer Terms of Service. If you are transacting under a reseller billing account, you cannot purchase Marketplace solutions. Contact your Google Cloud Partner Sales Manager for more information.

Your app will use compute instances managed in a logical grouping called a "cluster", which will be configured in a way that's great for getting started with Kubernetes. For more options, visit the Kubernetes engine cluster creation page.

Zone: us-central1-a

Network: default

Subnetwork: default

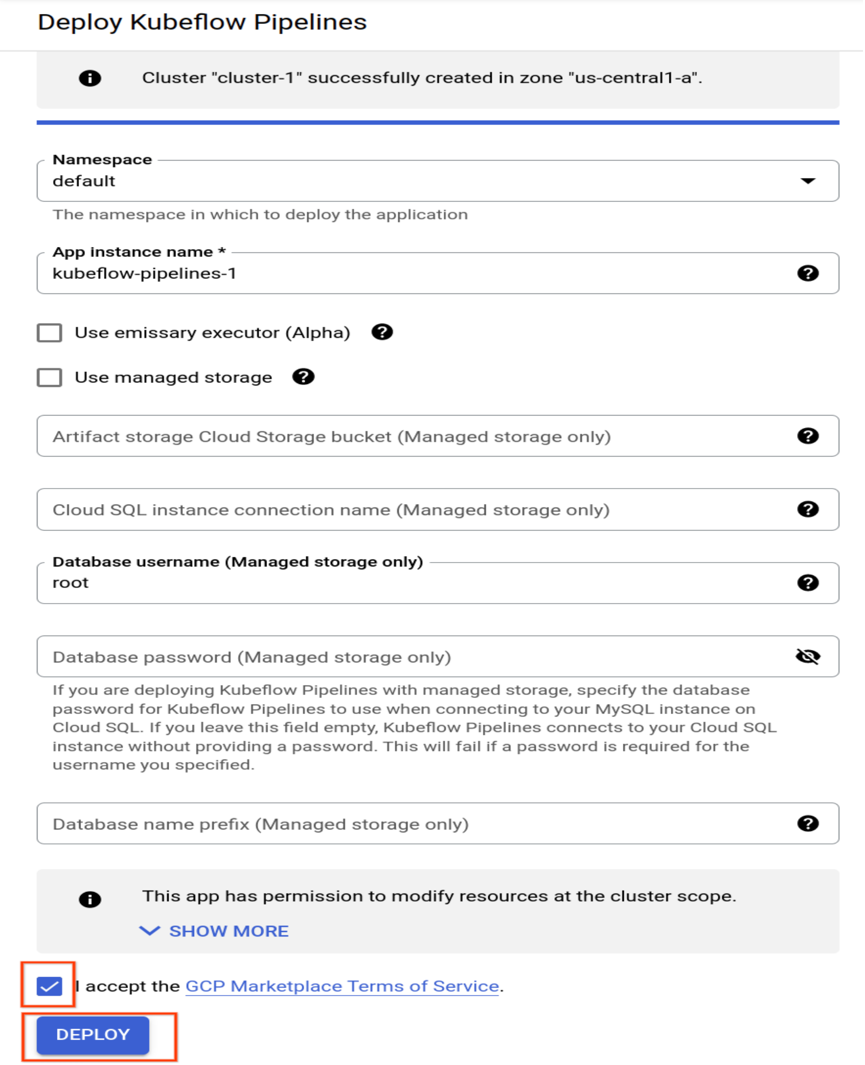
☑ Allow access to the following Cloud APIs
https://www.googleapis.com/auth/cloud-platform

Namespace: default
The namespace in which to deploy the application

App instance name *: kubeflow-pipelines-1

This should take 2-3 minutes to complete. Wait for the cluster to finish before proceeding to the next step. In the first tab opened, you can view the Cluster Creation taking place in the GKE section of the Cloud Console, or see the individual VMs spinning up in the GCE section of the Cloud Console.

6.When the cluster creation is complete, check the **Terms of Service** box, leave other settings unchanged, and then click **Deploy**. You will see the individual services of KFP deployed to your GKE cluster. Proceed to the next step while installation occurs.



Deploy Kubeflow Pipelines

Cluster "cluster-1" successfully created in zone "us-central1-a".

Namespace
default

The namespace in which to deploy the application

App instance name *
kubeflow-pipelines-1

□ Use emissary executor (Alpha)

□ Use managed storage

Artifact storage Cloud Storage bucket (Managed storage only)

Cloud SQL instance connection name (Managed storage only)

Database username (Managed storage only)
root

Database password (Managed storage only)

If you are deploying Kubeflow Pipelines with managed storage, specify the database password for Kubeflow Pipelines to use when connecting to your MySQL instance on Cloud SQL. If you leave this field empty, Kubeflow Pipelines connects to your Cloud SQL instance without providing a password. This will fail if a password is required for the username you specified.

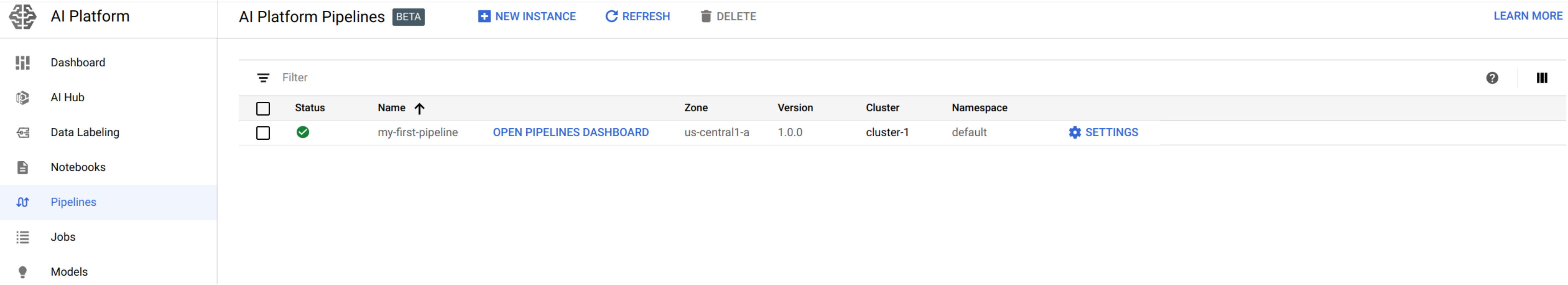
Database name prefix (Managed storage only)

This app has permission to modify resources at the cluster scope.

☑ I accept the GCP Marketplace Terms of Service.

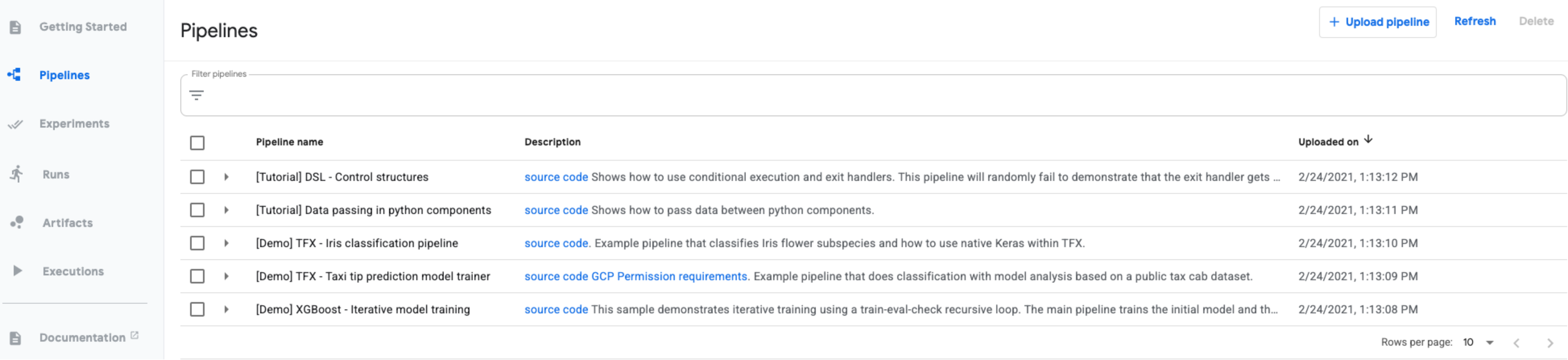
Task 2. Run an example pipeline

1.In the Google Cloud Console, on the Navigation menu, click **AI Platform** > **Pipelines**.
You will see the newly created Pipelines instance. If needed, click **Refresh** to update the page.
2.Click on the **OPEN PIPELINES DASHBOARD** link next to your instance name.



□	Status	Name ↑		Zone	Version	Cluster	Namespace	
□	✓	my-first-pipeline	OPEN PIPELINES DASHBOARD	us-central1-a	1.0.0	cluster-1	default	SETTINGS

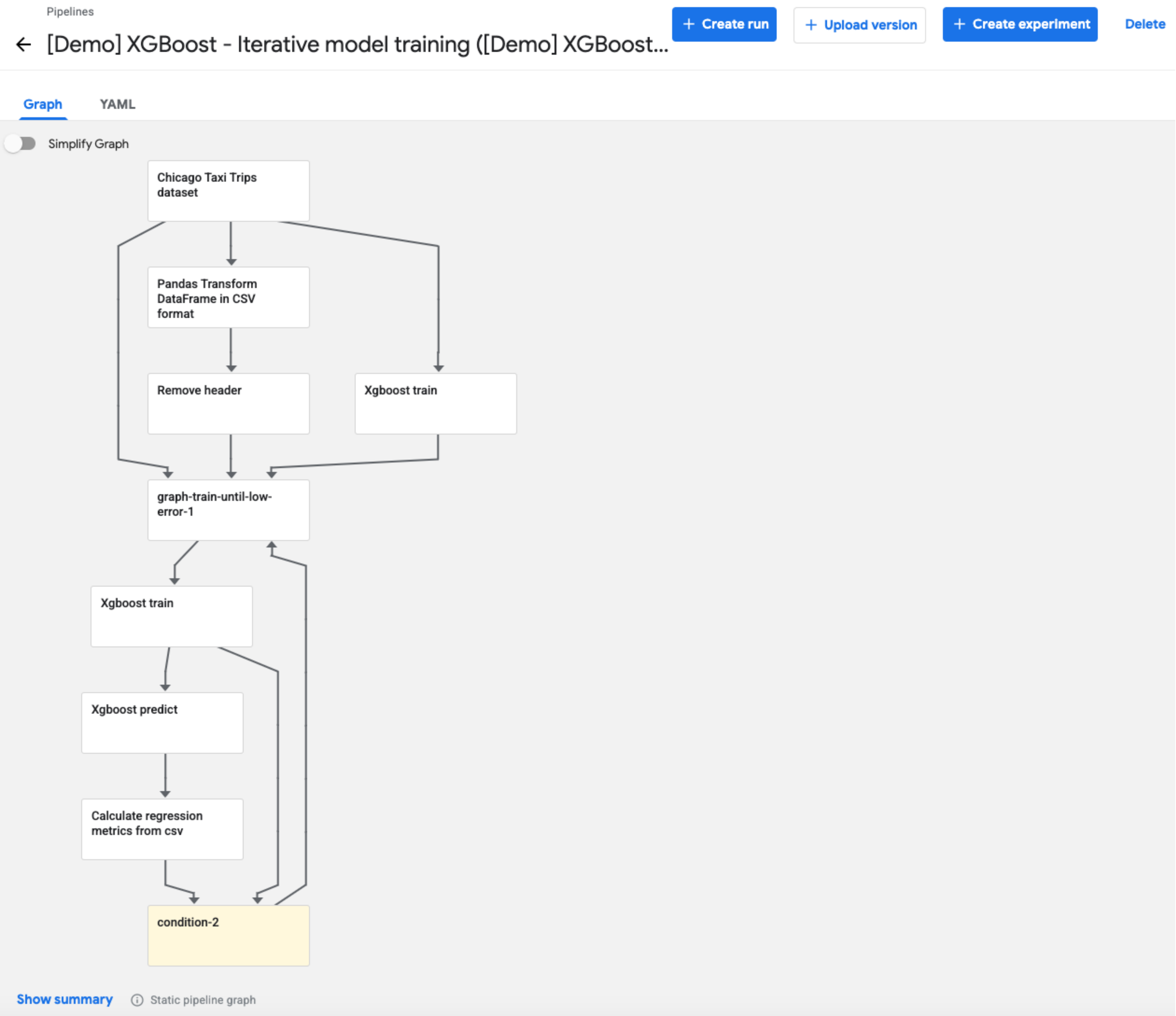
3.On the new page that loads, on the Navigation Menu on the left, click on **Pipelines**.
You will see a list of pipelines that have been provided for demo and tutorial purposes. For this lab, you will use the **[Demo] XGBoost - Iterative model training** sample pipeline. This sample demonstrates continuous training using a train-eval-check recursive loop, in which the model is trained iteratively until the model evaluation metrics are adequate.



Pipelines

□		Pipeline name	Description	Uploaded on ↓
□	▸	[Tutorial] DSL - Control structures	source code Shows how to use conditional execution and exit handlers. This pipeline will randomly fail to demonstrate that the exit handler gets ...	2/24/2021, 1:13:12 PM
□	▸	[Tutorial] Data passing in python components	source code Shows how to pass data between python components.	2/24/2021, 1:13:11 PM
□	▸	[Demo] TFX - Iris classification pipeline	source code. Example pipeline that classifies Iris flower subspecies and how to use native Keras within TFX.	2/24/2021, 1:13:10 PM
□	▸	[Demo] TFX - Taxi tip prediction model trainer	source code GCP Permission requirements. Example pipeline that does classification with model analysis based on a public tax cab dataset.	2/24/2021, 1:13:09 PM
□	▸	[Demo] XGBoost - Iterative model training	source code This sample demonstrates iterative training using a train-eval-check recursive loop. The main pipeline trains the initial model and th...	2/24/2021, 1:13:08 PM

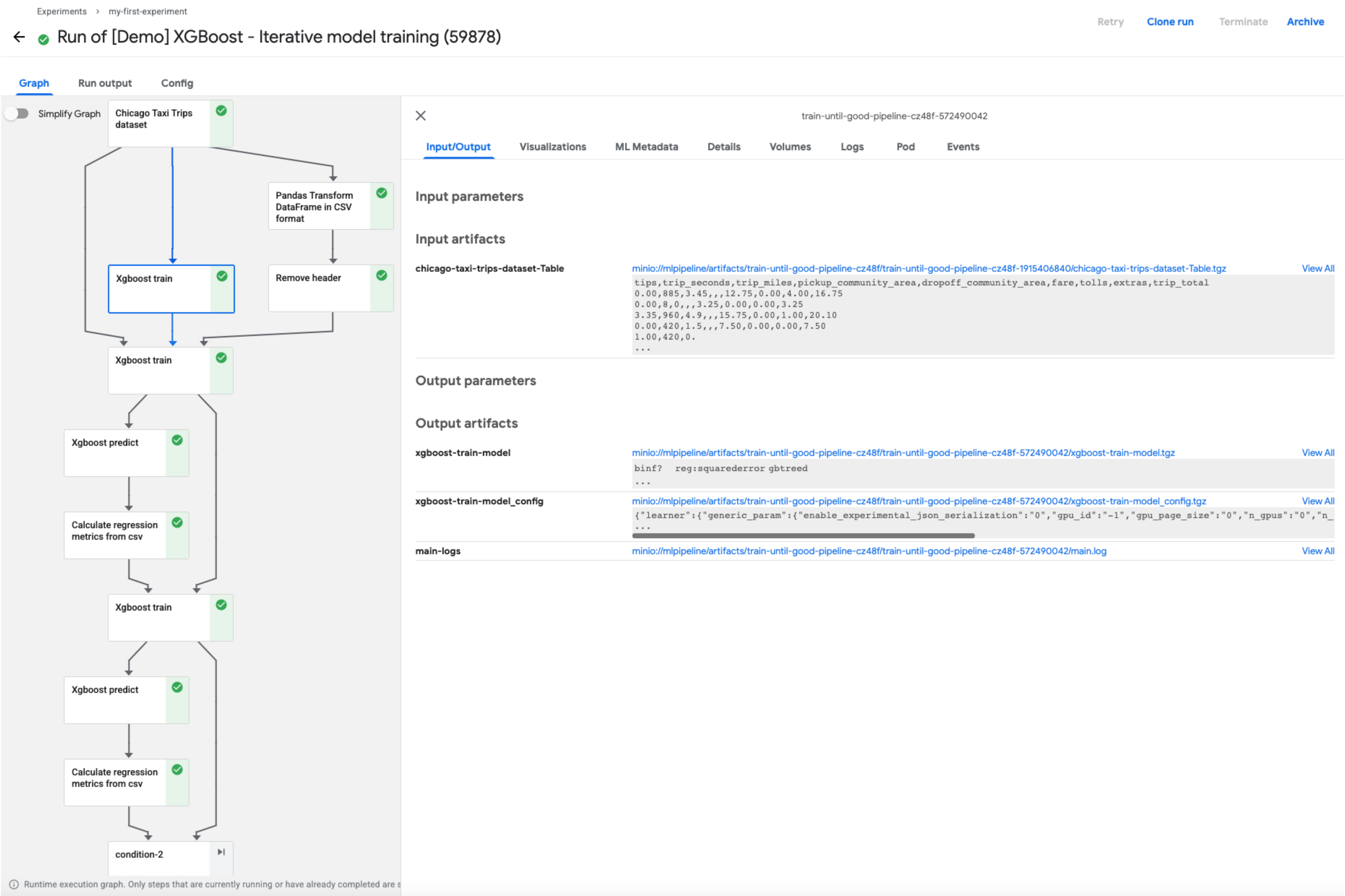
4.Click on the **[Demo] XGBoost - Iterative model training** pipeline.
When it loads, you can see what the graph for this pipeline looks like. Next, you will create a run to test this pipeline.



Pipelines

[Demo] XGBoost - Iterative model training ([Demo] XGBoost...

5.Click on **Create experiment** on the top right to associate a new experiment for the run.
6.Enter the name **my-first-experiment** in the form that loads, and then click **Next**.
7.Leave the default options, and click **Start** to run the pipeline. The pipeline run may take a few minutes to complete. You can click **Refresh** to update the page and see the latest status.
8.Once the pipeline run has finished, you can click on the run name to see the fully generated graph as well as performance metrics and graphs.
The green check marks means every part of the pipeline ran successfully. You can click on any box and see the outputs for that part like input/output, visualizations, logs, events, etc.



Run of [Demo] XGBoost - Iterative model training (59878)

train-until-good-pipeline-cz48f-572490042

Input parameters

Input artifacts

chicago-taxi-trips-dataset-Table minio://mlpipeline/artifacts/train-until-good-pipeline-cz48f/train-until-good-pipeline-cz48f-1915406840/chicago-taxi-trips-dataset-Table.tgz View All

```
tips,trip_seconds,trip_miles,pickup_community_area,dropoff_community_area,fare,tolls,extras,trip_total
0.00,885,3.45,,,12.75,0.00,4.00,16.75
0.00,8,0,,,3.25,0.00,0.00,3.25
3.35,960,4.9,,,15.75,0.00,1.00,20.10
0.00,420,1.5,,,7.50,0.00,0.00,7.50
1.00,420,0.
...
```

Output parameters

Output artifacts

xgboost-train-model minio://mlpipeline/artifacts/train-until-good-pipeline-cz48f/train-until-good-pipeline-cz48f-572490042/xgboost-train-model.tgz View All

```
binf?  reg:squarederror gbtreed
...
```

xgboost-train-model_config minio://mlpipeline/artifacts/train-until-good-pipeline-cz48f/train-until-good-pipeline-cz48f-572490042/xgboost-train-model_config.tgz View All

```
{"learner":{"generic_param":{"enable_experimental_json_serialization":"0","gpu_id":"-1","gpu_page_size":"0","n_gpus":"0","n_
...
```

main-logs minio://mlpipeline/artifacts/train-until-good-pipeline-cz48f/train-until-good-pipeline-cz48f-572490042/main.log View All

ⓘ Runtime execution graph. Only steps that are currently running or have already completed are s