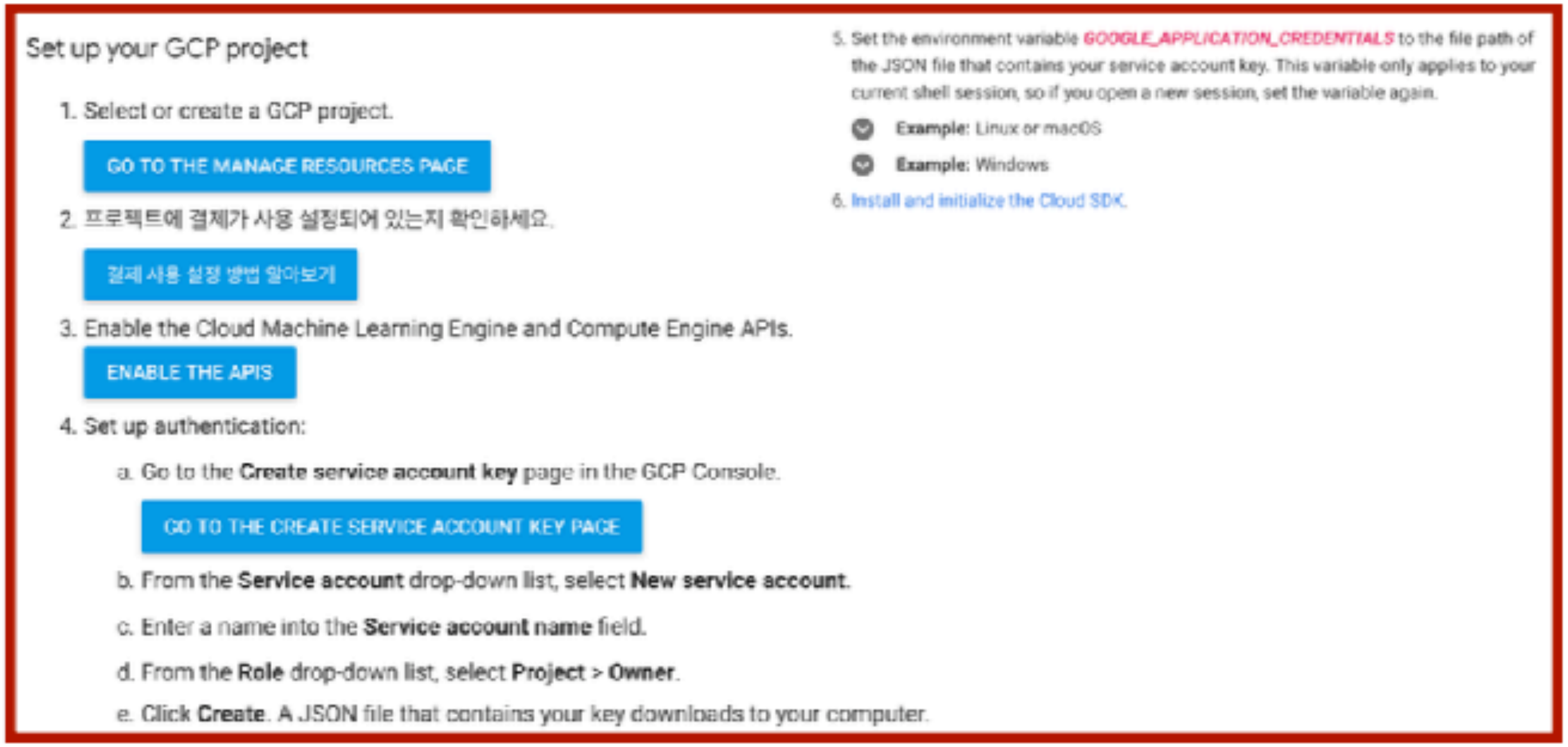
Cloud ML Engine

Google Cloud Platform (GCP)

2017.08
박찬성

before getting started..

1. do you have a GCP account? if no, just create one
    1. if you have a gmail account, you are ok to go
2. just read THIS, it will give a basic idea how everything works. You don't need to follow every instructions. just reading is sufficient
3. do some of the pre-requisite steps introduced like..



Set up your GCP project

1. Select or create a GCP project.

GO TO THE MANAGE RESOURCES PAGE

2. 프로젝트에 결제가 사용 설정되어 있는지 확인하세요.

결제 사용 설정 방법 알아보기

3. Enable the Cloud Machine Learning Engine and Compute Engine APIs.

ENABLE THE APIS

4. Set up authentication:

a. Go to the **Create service account key** page in the GCP Console.

GO TO THE CREATE SERVICE ACCOUNT KEY PAGE

b. From the **Service account** drop-down list, select **New service account**.

c. Enter a name into the **Service account name** field.

d. From the **Role** drop-down list, select **Project** > **Owner**.

e. Click **Create**. A JSON file that contains your key downloads to your computer.

5. Set the environment variable *GOOGLE_APPLICATION_CREDENTIALS* to the file path of the JSON file that contains your service account key. This variable only applies to your current shell session, so if you open a new session, set the variable again.

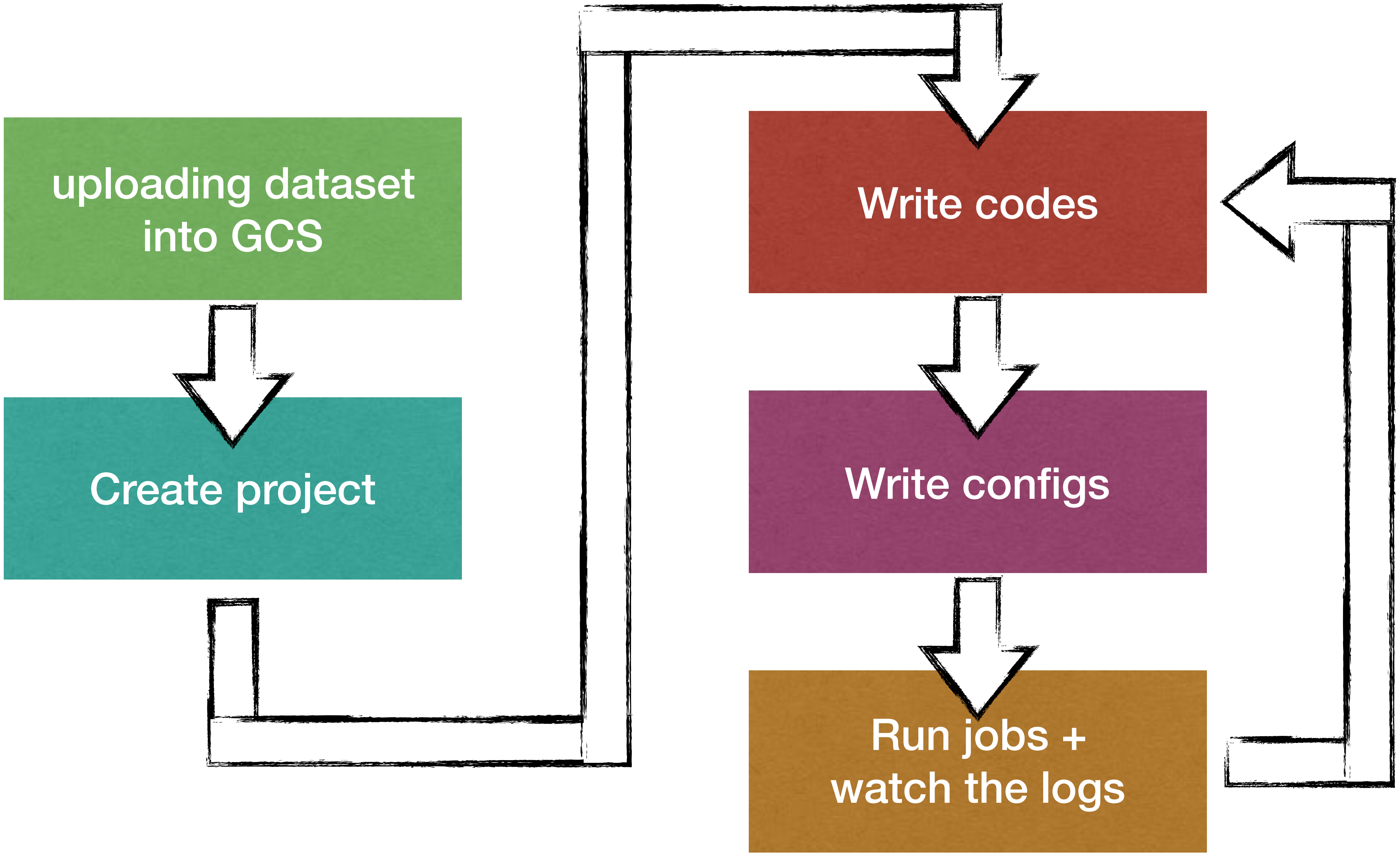
Example: Linux or macOS

Example: Windows

6. Install and initialize the Cloud SDK.

basic flow introduction

1. create a bucket, and some folders in it
2. upload files to folders
3. create a project source tree
4. create a setup.py if you have any dependencies to install
5. create a config.yaml for your configuration of the project
6. create an empty __init__.py in every other folders except for the root path
7. write some codes
8. or copy some code files from local
    1. change print() function to logging to see a real-time logs
9. submit a job
10. watch the logs



uploading dataset into GCS

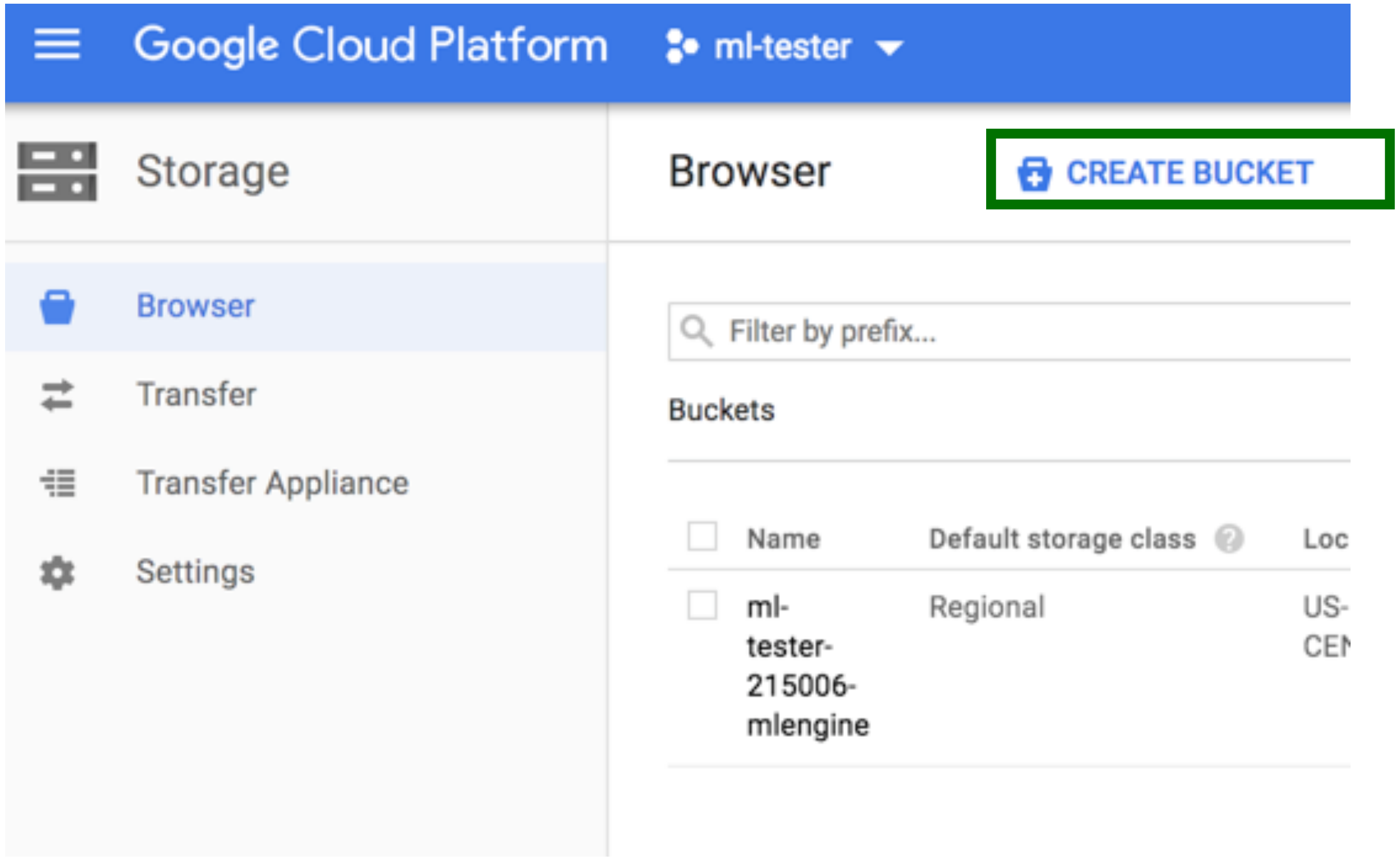
Create project

Write codes

Write configs

Run jobs + watch the logs

create GCS bucket

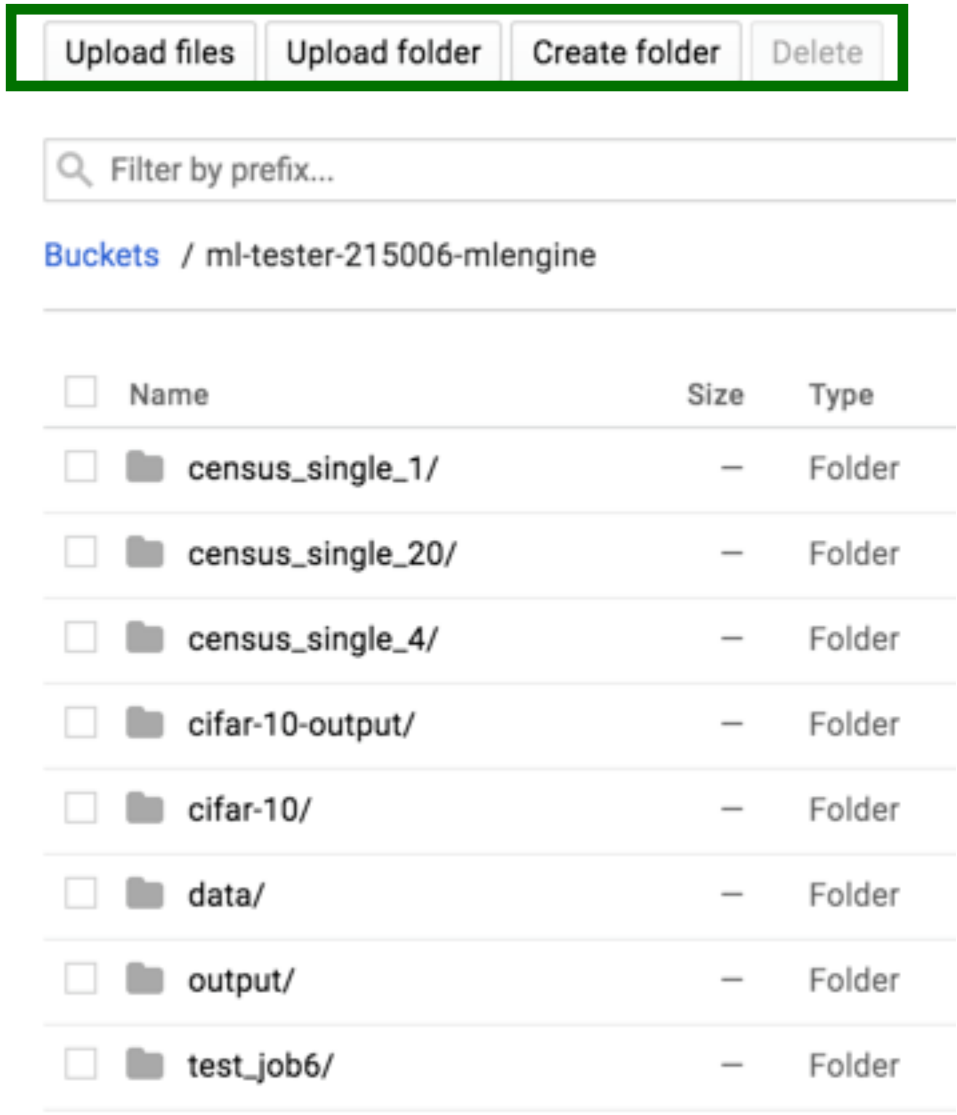


□	Name	Default storage class	Loc
□	ml-tester-215006-mlengine	Regional	US-CEN

make a bucket
@ where cloud ML engine is available

bucket and cloud ML trainer should be placed @the same location

Create folder + Upload files



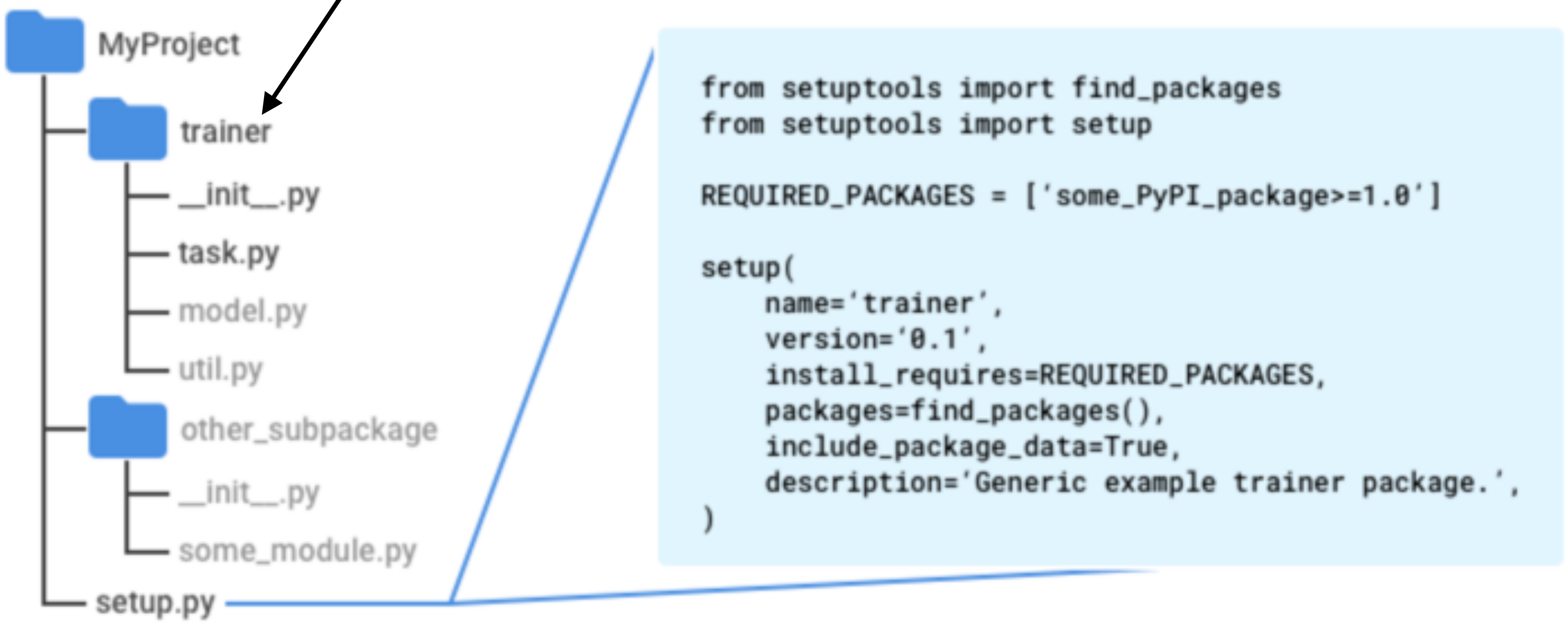
Buckets / ml-tester-215006-mlengine

Name	Size	Type
census_single_1/	–	Folder
census_single_20/	–	Folder
census_single_4/	–	Folder
cifar-10-output/	–	Folder
cifar-10/	–	Folder
data/	–	Folder
output/	–	Folder
test_job6/	–	Folder

SUPER EASY

Project Structure

main application module



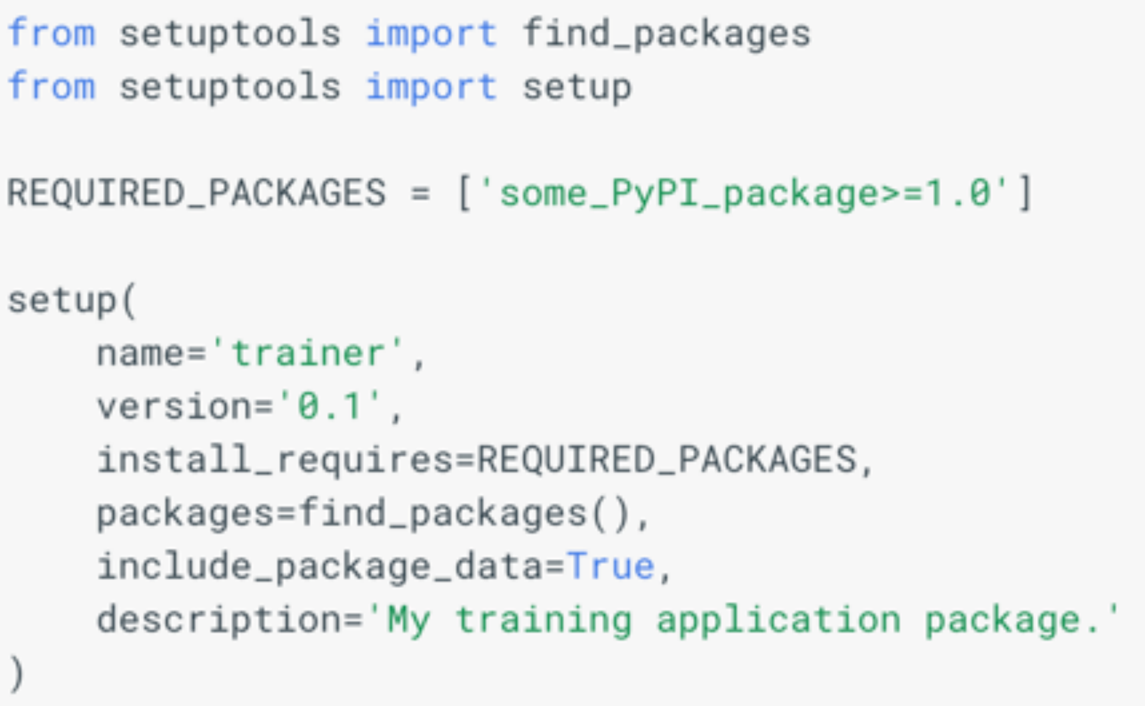
```
from setuptools import find_packages
from setuptools import setup

REQUIRED_PACKAGES = ['some_PyPI_package>=1.0']

setup(
    name='trainer',
    version='0.1',
    install_requires=REQUIRED_PACKAGES,
    packages=find_packages(),
    include_package_data=True,
    description='Generic example trainer package.',
)
```

Create an `__init__.py` file in every subdirectory. These files are used by Setuptools to identify directories with code to package, and may be empty

Dependencies

- runtime versions shows packages that are already installed. You need to specify which version to use.
  * --runtime_version x.x

- standard dependencies can be defined in setup.py file



```
from setuptools import find_packages
from setuptools import setup

REQUIRED_PACKAGES = ['some_PyPI_package>=1.0']

setup(
    name='trainer',
    version='0.1',
    install_requires=REQUIRED_PACKAGES,
    packages=find_packages(),
    include_package_data=True,
    description='My training application package.'
)
```

find packages
of your interest
@PyPI

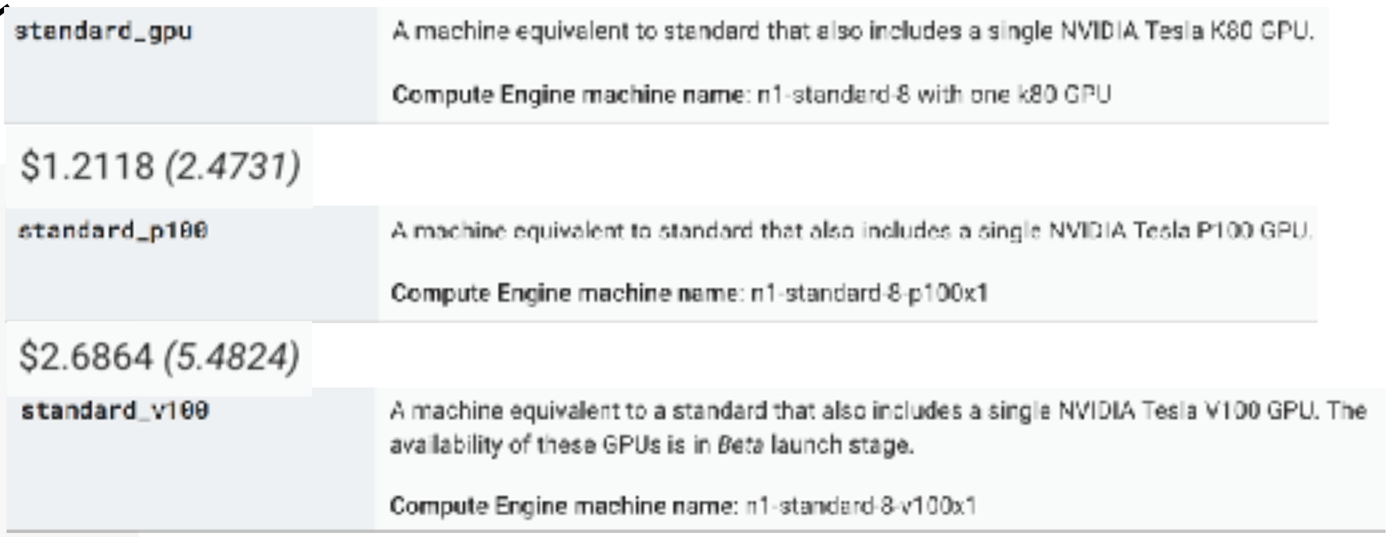
gcloud cli will automatically recognize the setup.py

configuration (1)

set option “--config" to specify the location of config.yaml

```
--config=config.yaml
```

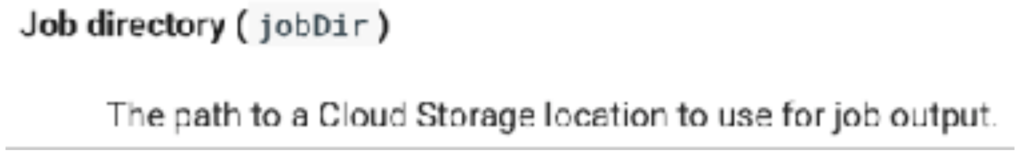
see the machine types, and here



standard_gpu	A machine equivalent to standard that also includes a single NVIDIA Tesla K80 GPU. Compute Engine machine name: n1-standard-8 with one k80 GPU
$1.2118 *(2.4731)*	
standard_p100	A machine equivalent to standard that also includes a single NVIDIA Tesla P100 GPU. Compute Engine machine name: n1-standard-8-p100x1
$2.6864 *(5.4824)*	
standard_v100	A machine equivalent to a standard that also includes a single NVIDIA Tesla V100 GPU. The availability of these GPUs is in *Beta* launch stage. Compute Engine machine name: n1-standard-8-v100x1

config.yaml example

```
trainingInput:
  scaleTier: CUSTOM
  masterType: complex_model_m
  workerType: complex_model_m
  parameterServerType: large_model
  workerCount: 9
  parameterServerCount: 3
  packageUris: gs://my/trainer/path/package-0.0.0.tar.gz
  pythonModule: trainer.task
  region: us-central1
  jobDir: gs://my/training/job/directory
  runtimeVersion: 1.10
  pythonVersion: 3.5
```

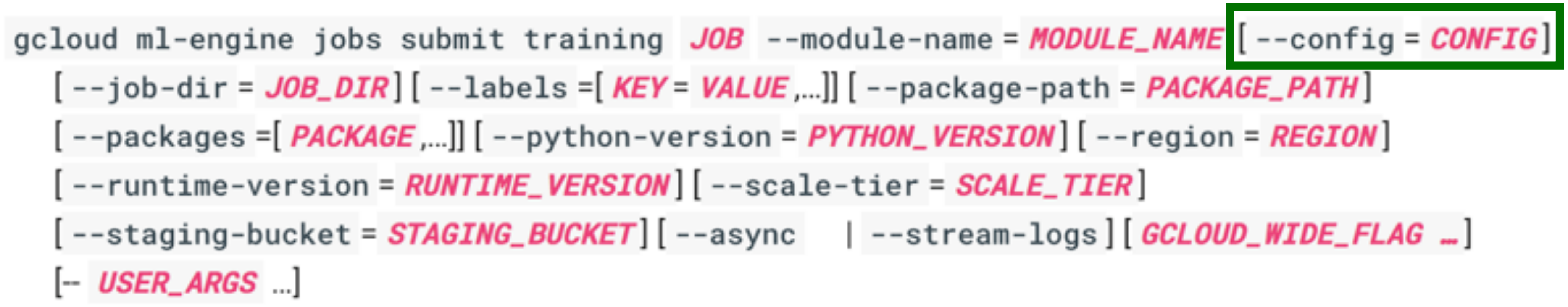


Job directory (`jobDir`)

The path to a Cloud Storage location to use for job output.

configuration (2)

other common training parameters



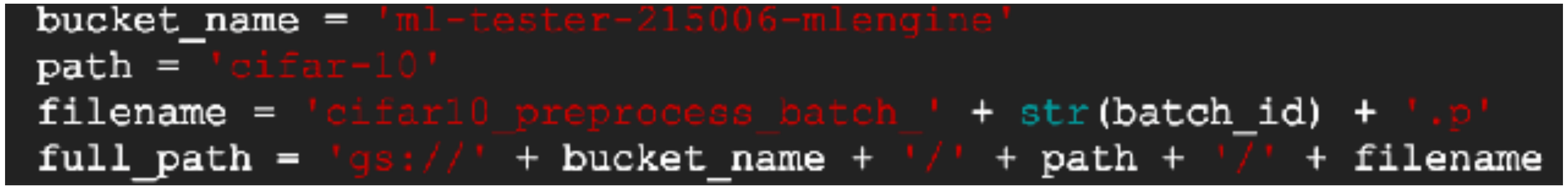
```
gcloud ml-engine jobs submit training JOB --module-name = MODULE_NAME [--config = CONFIG]
   [--job-dir = JOB_DIR] [--labels =[ KEY = VALUE ,…]] [--package-path = PACKAGE_PATH]
   [--packages =[ PACKAGE ,…]] [--python-version = PYTHON_VERSION] [--region = REGION]
   [--runtime-version = RUNTIME_VERSION] [--scale-tier = SCALE_TIER]
   [--staging-bucket = STAGING_BUCKET] [--async   | --stream-logs] [ GCLOUD_WIDE_FLAG …]
   [-- USER_ARGS …]
```

further description for each parameter

read/write file from GCS?

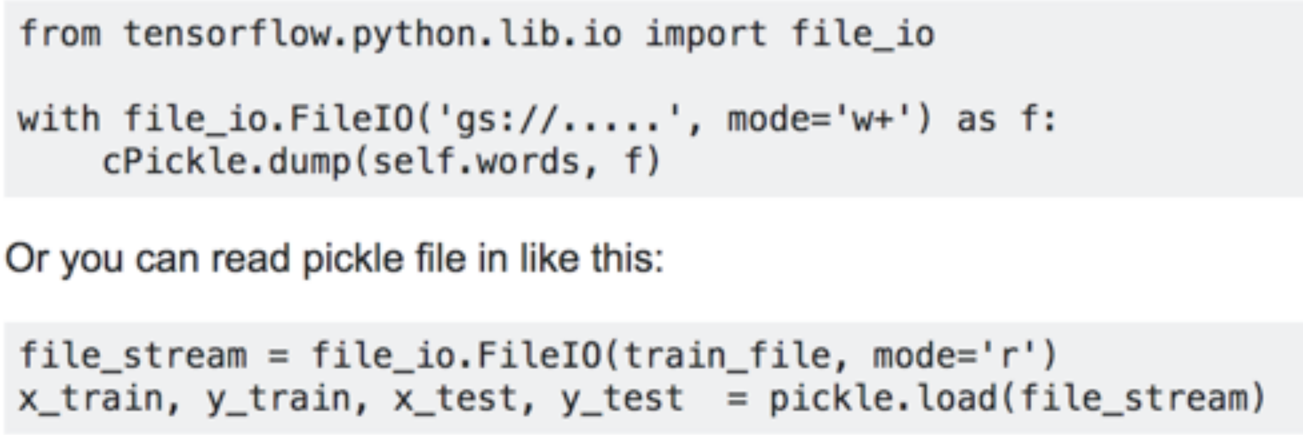
from tensorflow.python.lib.io **import** file_io

python standard library can’t recognize GCS location,
but, tensorflow provides file_io packages for it



```
bucket_name = 'ml-tester-215006-mlengine'
path = 'cifar-10'
filename = 'cifar10_preprocess_batch_' + str(batch_id) + '.p'
full_path = 'gs://' + bucket_name + '/' + path + '/' + filename
```

GCS location can be recognized by three parts,
“gs://” protocol, “bucket_name”, and “file_path”



```
from tensorflow.python.lib.io import file_io

with file_io.FileIO('gs://.....', mode='w+') as f:
    cPickle.dump(self.words, f)
```

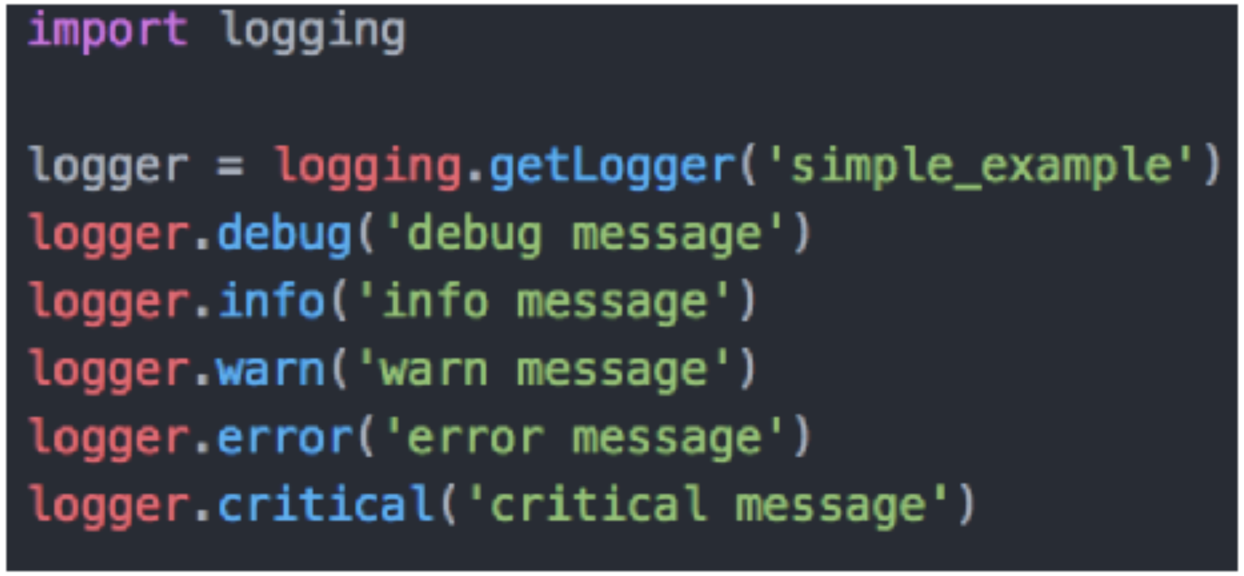
Or you can read pickle file in like this:

```
file_stream = file_io.FileIO(train_file, mode='r')
x_train, y_train, x_test, y_test  = pickle.load(file_stream)
```

logging

- standard python’s print function shows up “AFTER” everything is done. This is a known buffering issue.

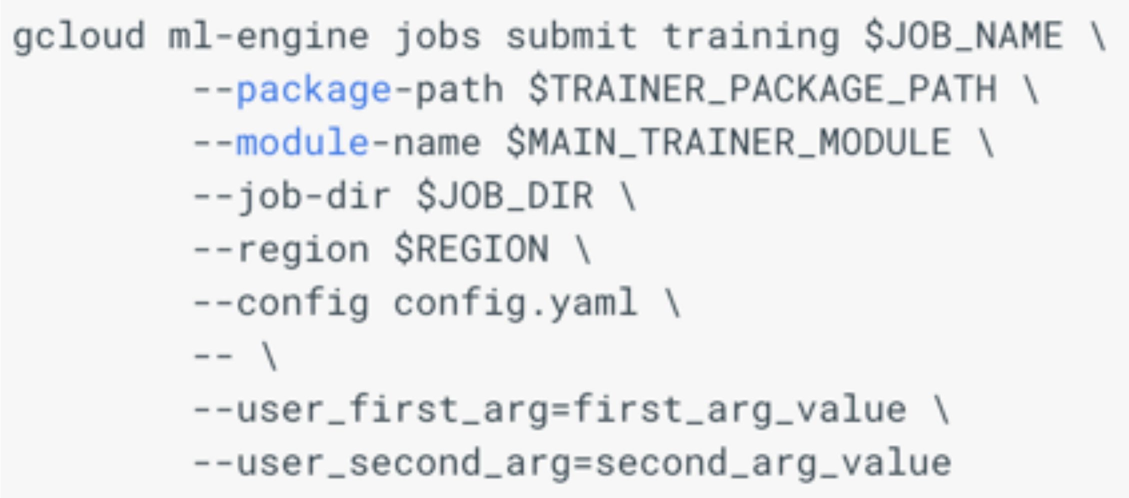
- If you want to check a real time logs, you need to use “logging” package instead.



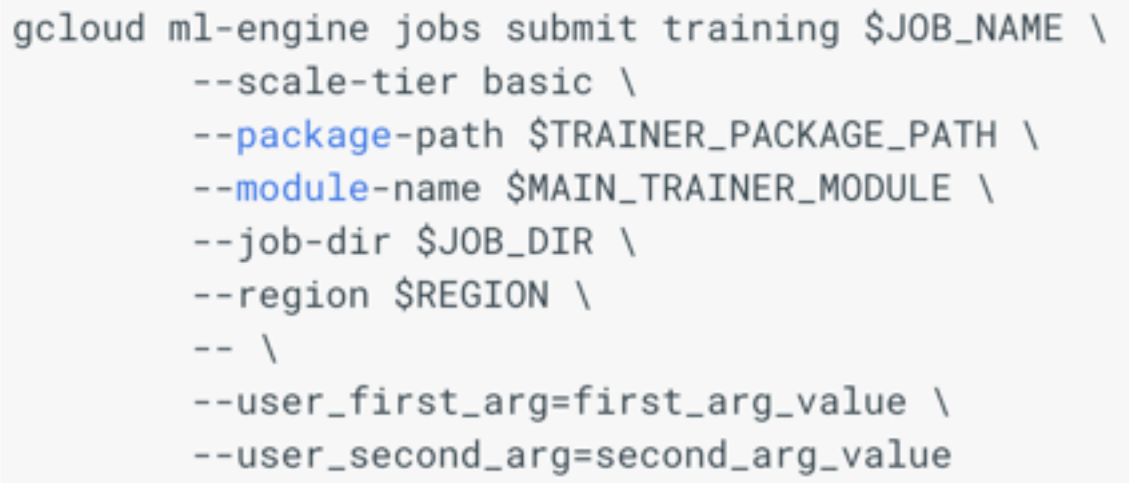
```
import logging

logger = logging.getLogger('simple_example')
logger.debug('debug message')
logger.info('info message')
logger.warn('warn message')
logger.error('error message')
logger.critical('critical message')
```

submitting a job (1)

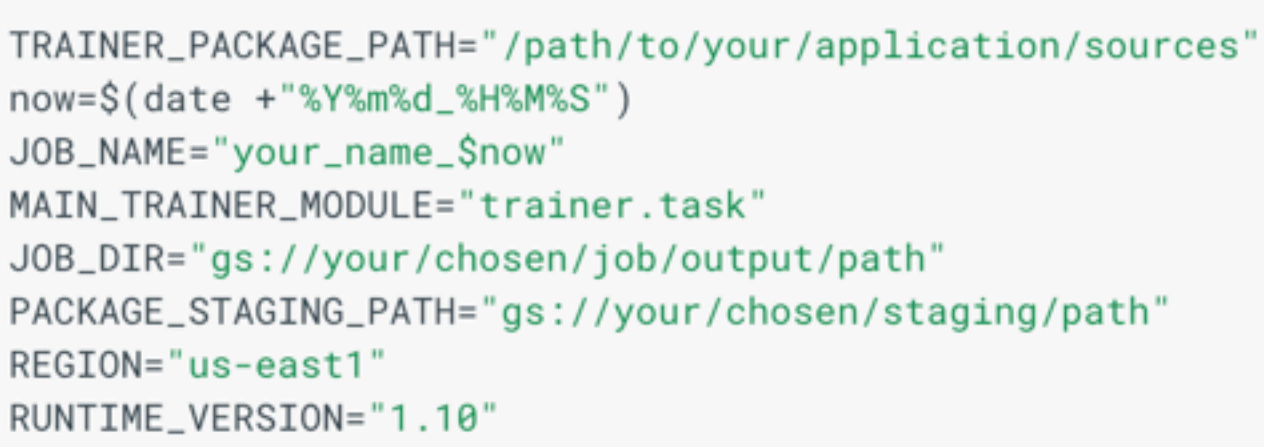
with common parameters in option flags



```
gcloud ml-engine jobs submit training $JOB_NAME \
        --package-path $TRAINER_PACKAGE_PATH \
        --module-name $MAIN_TRAINER_MODULE \
        --job-dir $JOB_DIR \
        --region $REGION \
        --config config.yaml \
        -- \
        --user_first_arg=first_arg_value \
        --user_second_arg=second_arg_value
```



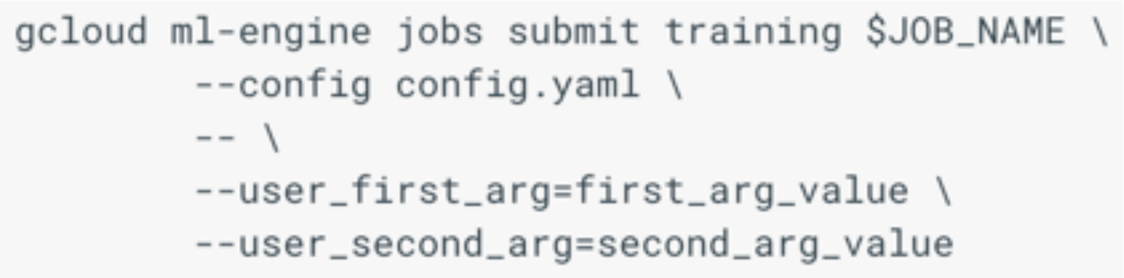
```
gcloud ml-engine jobs submit training $JOB_NAME \
        --scale-tier basic \
        --package-path $TRAINER_PACKAGE_PATH \
        --module-name $MAIN_TRAINER_MODULE \
        --job-dir $JOB_DIR \
        --region $REGION \
        -- \
        --user_first_arg=first_arg_value \
        --user_second_arg=second_arg_value
```



```
TRAINER_PACKAGE_PATH="/path/to/your/application/sources"
now=$(date +"%Y%m%d_%H%M%S")
JOB_NAME="your_name_$now"
MAIN_TRAINER_MODULE="trainer.task"
JOB_DIR="gs://your/chosen/job/output/path"
PACKAGE_STAGING_PATH="gs://your/chosen/staging/path"
REGION="us-east1"
RUNTIME_VERSION="1.10"
```

submitting a job (2)

with only config.yaml configuration file,
in which every parameters are defined

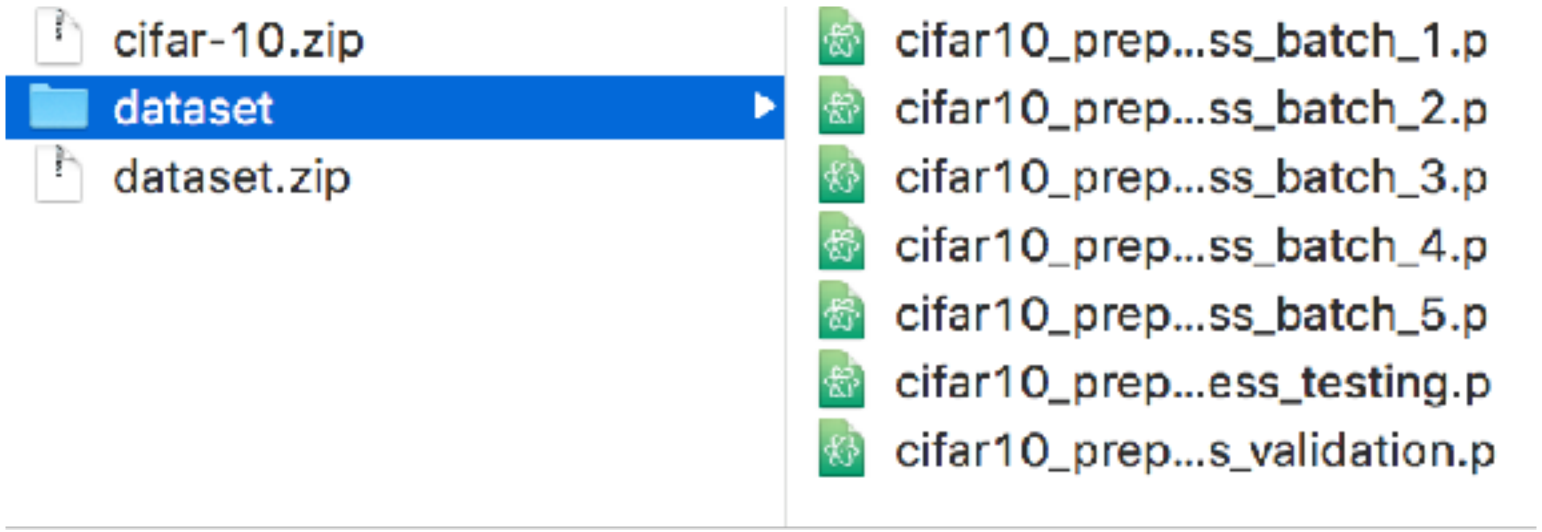


```
gcloud ml-engine jobs submit training $JOB_NAME \
        --config config.yaml \
        -- \
        --user_first_arg=first_arg_value \
        --user_second_arg=second_arg_value
```

Example

- showing a simple example of CIFAR10 image classification with GoogLeNet (InceptionV1) model.
- This example will use TESLA V100/P100
- All the materials needed for this example can be downloaded from HERE

Step1 - extract materials



cifar-10.zip
dataset
dataset.zip

cifar10_prep...ss_batch_1.p
cifar10_prep...ss_batch_2.p
cifar10_prep...ss_batch_3.p
cifar10_prep...ss_batch_4.p
cifar10_prep...ss_batch_5.p
cifar10_prep...ess_testing.p
cifar10_prep...s_validation.p

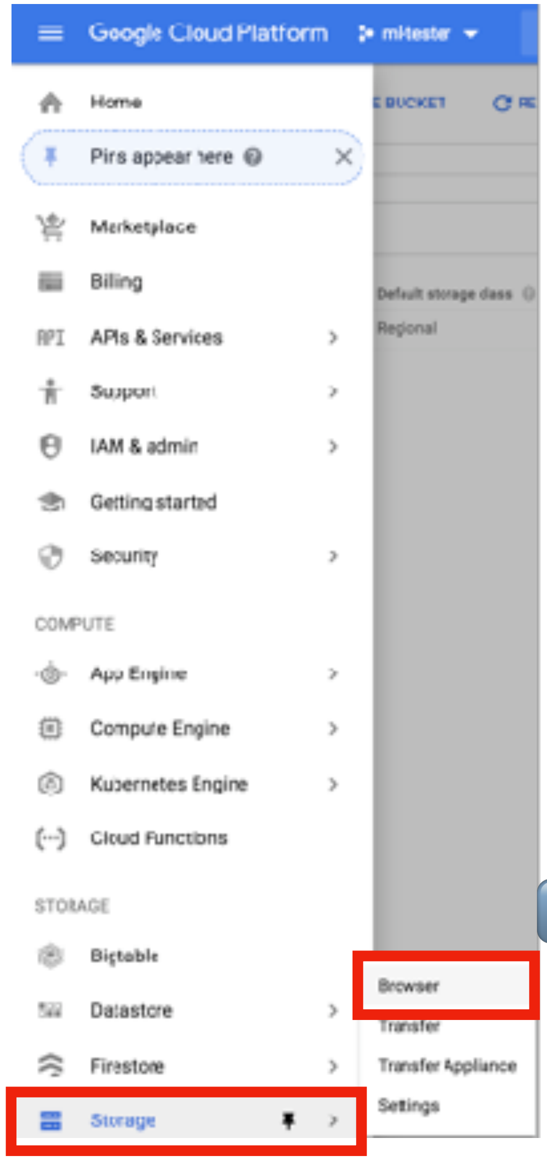
cifar-10 file contains source codes to train

dataset folder contains dataset to train on.
these files should be uploaded into the GCP bucket

Step2 - upload data to GCP

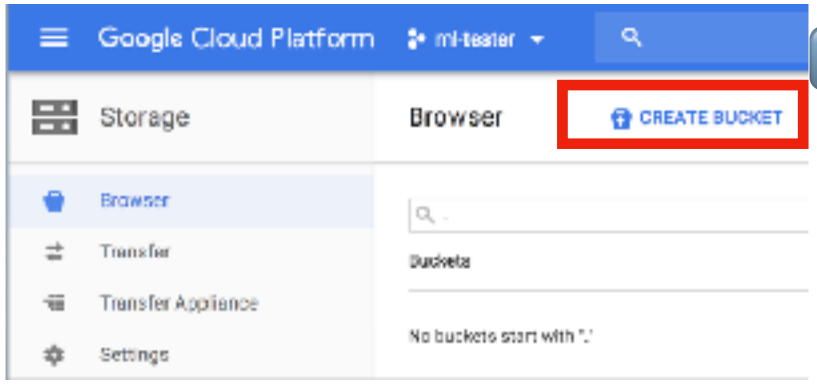
Also possible to download dataset from source code and save it to the bucket

here, I will show how to upload manually though

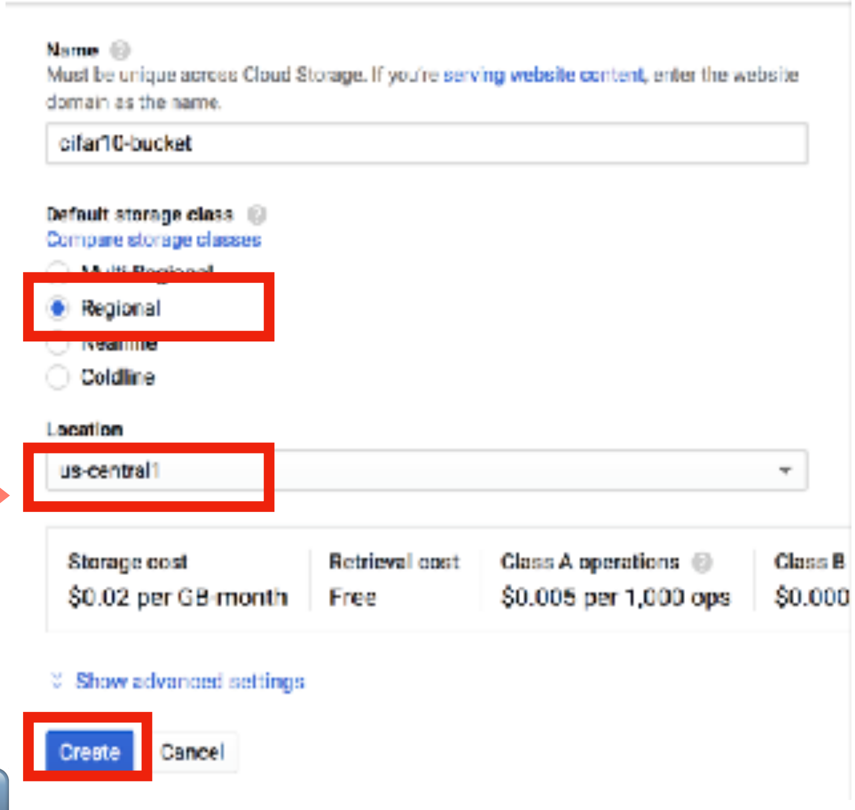


1

2



3



Name
Must be unique across Cloud Storage. If you're serving website content, enter the website domain as the name.

cifar10-bucket

Default storage class
Compare storage classes

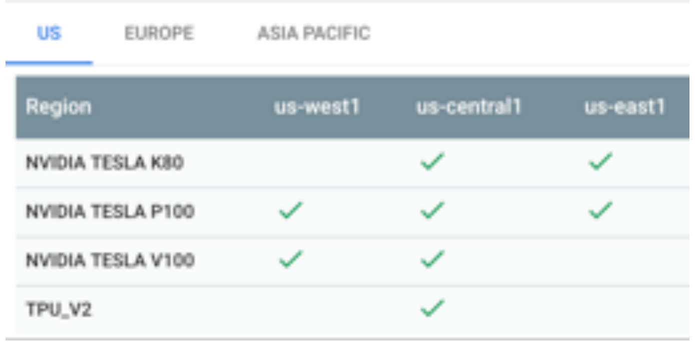
○ Multi-Regional
● Regional
○ Nearline
○ Coldline

Location
us-central1

Storage cost	Retrieval cost	Class A operations	Class B
$0.02 per GB-month	Free	$0.005 per 1,000 ops	$0.000

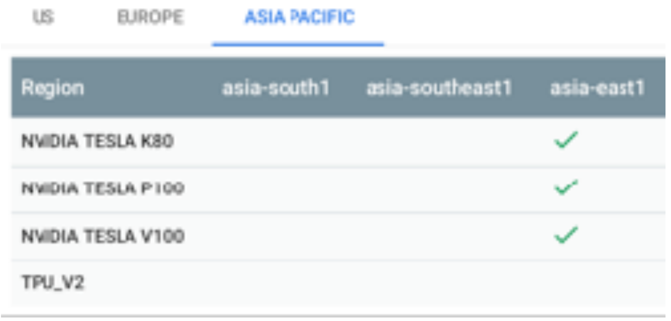
4

locations supporting GPUs/TPU



US | EUROPE | ASIA PACIFIC

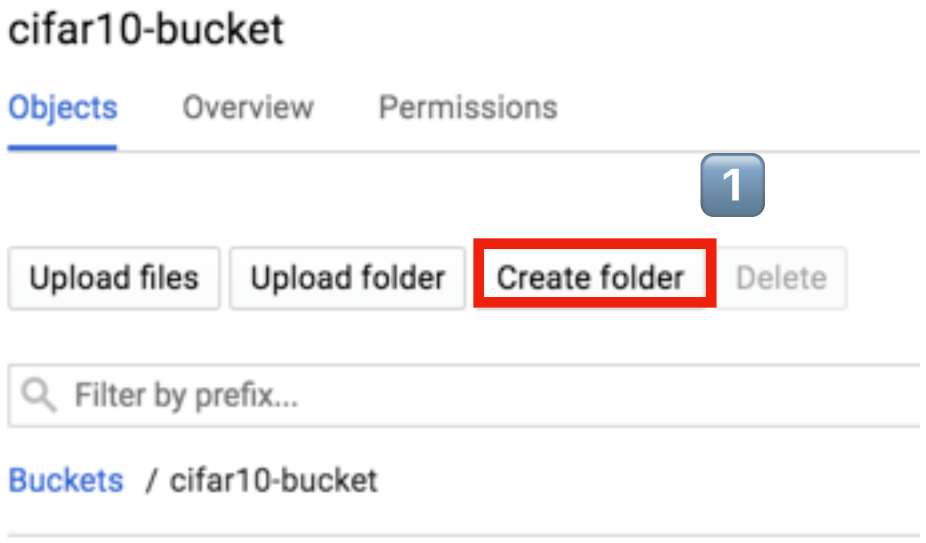
Region	us-west1	us-central1	us-east1
NVIDIA TESLA K80		✓	✓
NVIDIA TESLA P100	✓	✓	✓
NVIDIA TESLA V100	✓	✓	
TPU_V2		✓	



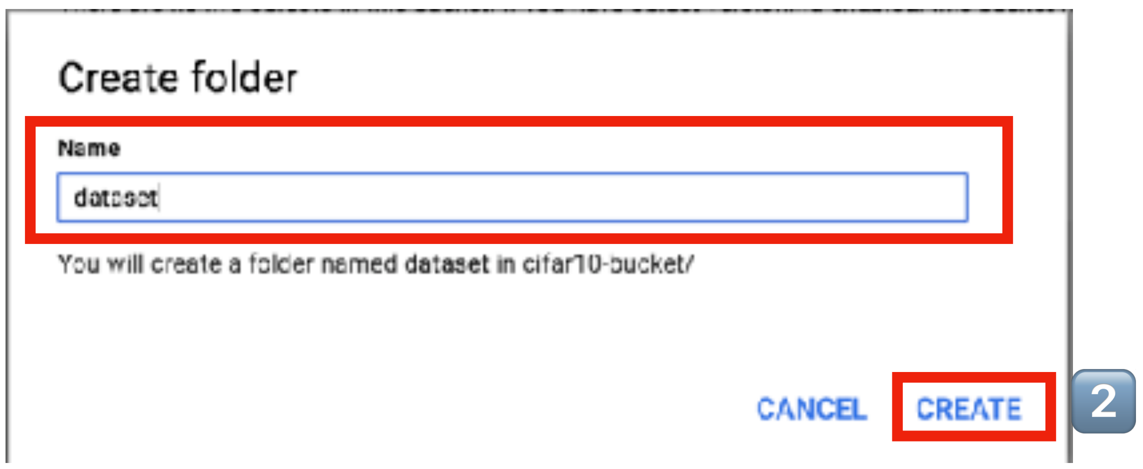
US | EUROPE | ASIA PACIFIC

Region	asia-south1	asia-southeast1	asia-east1
NVIDIA TESLA K80			✓
NVIDIA TESLA P100			✓
NVIDIA TESLA V100			✓
TPU_V2			

Step2 - upload data to GCP(2)



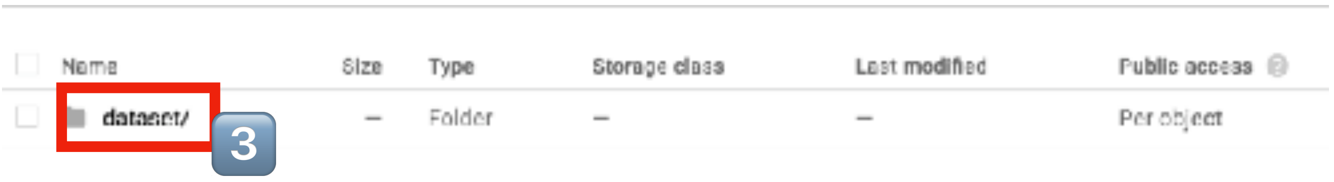
create folder



Name: dataset

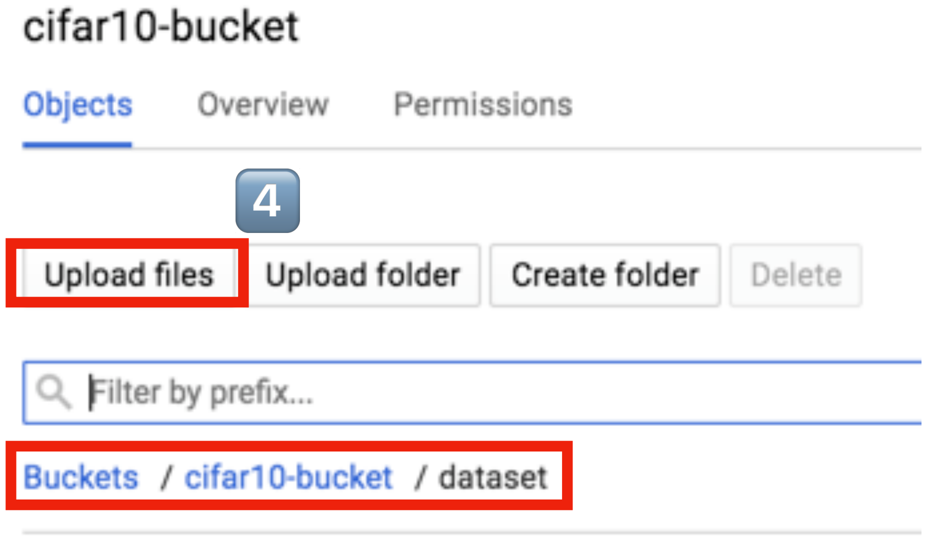
You will create a folder named dataset in cifar10-bucket/

name folder



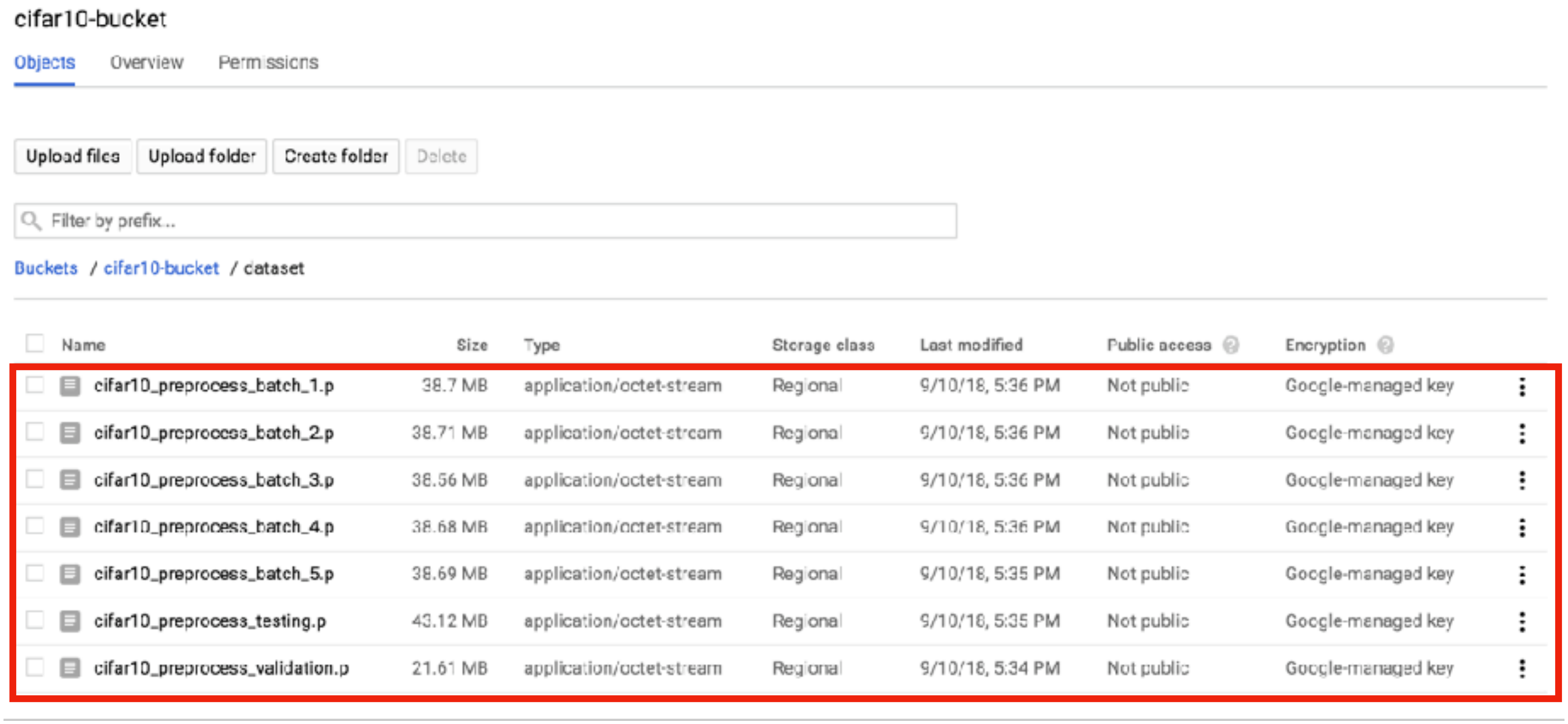
Name	Size	Type	Storage class	Last modified	Public access
dataset/	–	Folder	–	–	Per object

check if folder has been created successfully, and click the folder



upload dataset files into the folder

Step2 - upload data to GCP(3)

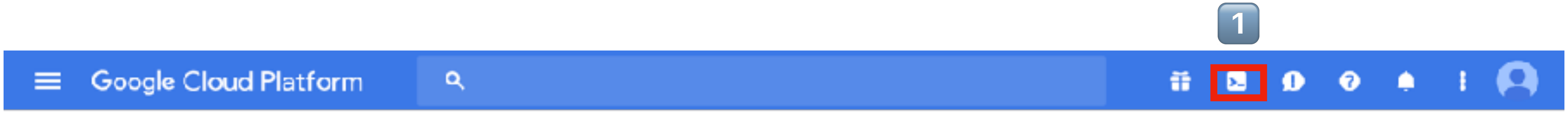


Name	Size	Type	Storage class	Last modified	Public access	Encryption
cifar10_preprocess_batch_1.p	38.7 MB	application/octet-stream	Regional	9/10/18, 5:36 PM	Not public	Google-managed key
cifar10_preprocess_batch_2.p	38.71 MB	application/octet-stream	Regional	9/10/18, 5:36 PM	Not public	Google-managed key
cifar10_preprocess_batch_3.p	38.56 MB	application/octet-stream	Regional	9/10/18, 5:36 PM	Not public	Google-managed key
cifar10_preprocess_batch_4.p	38.68 MB	application/octet-stream	Regional	9/10/18, 5:36 PM	Not public	Google-managed key
cifar10_preprocess_batch_5.p	38.69 MB	application/octet-stream	Regional	9/10/18, 5:35 PM	Not public	Google-managed key
cifar10_preprocess_testing.p	43.12 MB	application/octet-stream	Regional	9/10/18, 5:35 PM	Not public	Google-managed key
cifar10_preprocess_validation.p	21.61 MB	application/octet-stream	Regional	9/10/18, 5:34 PM	Not public	Google-managed key

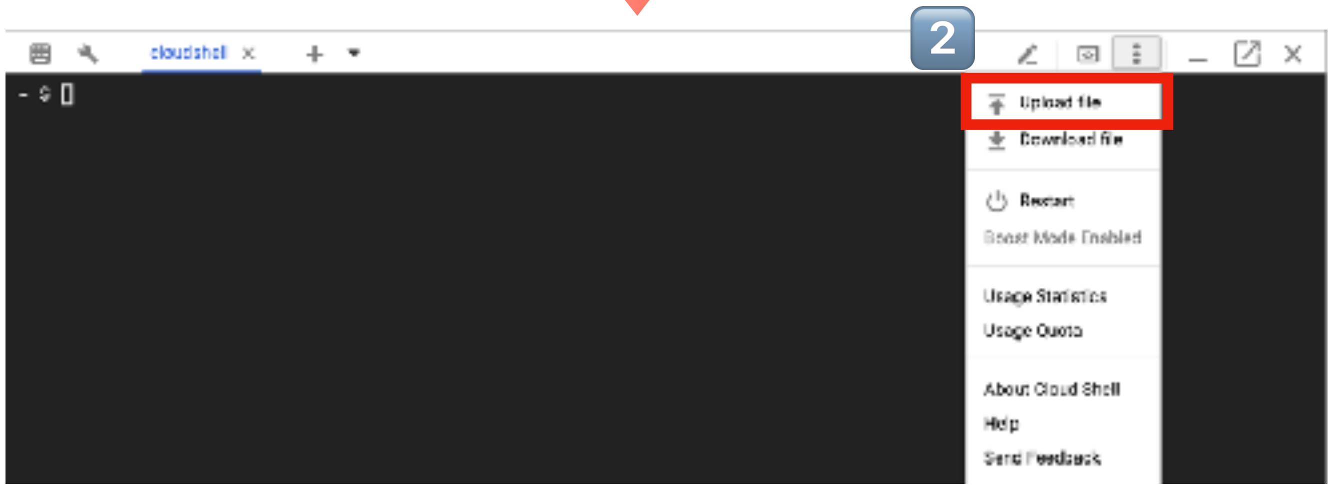
create “ckpt” folder just like “dataset” folder

- “ckpt” folder will store training checkpoint files

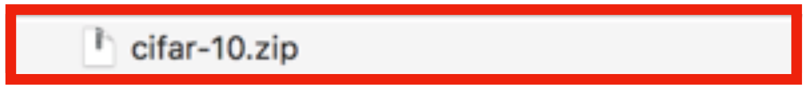
Step3 - upload source files



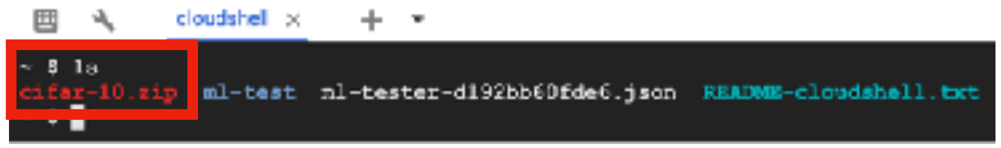
activate cloud shell



```
~ $ ▯
```



upload cifar-10.zip

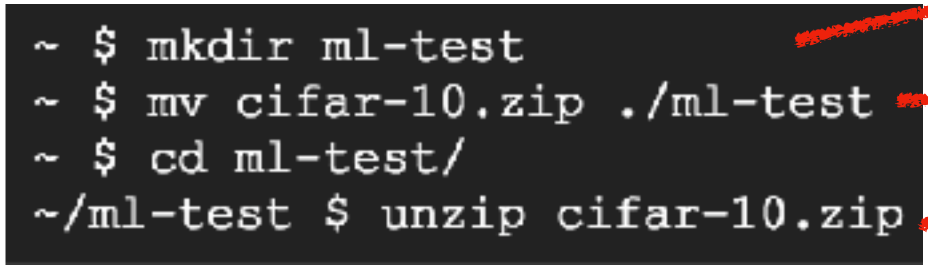


```
~ $ ls
cifar-10.zip  ml-test  ml-tester-d192bb60fde6.json  README-cloudshell.txt
~ $ ▮
```

check if the file exists

* uploading files will store files only under the home directory

step4-extract source files

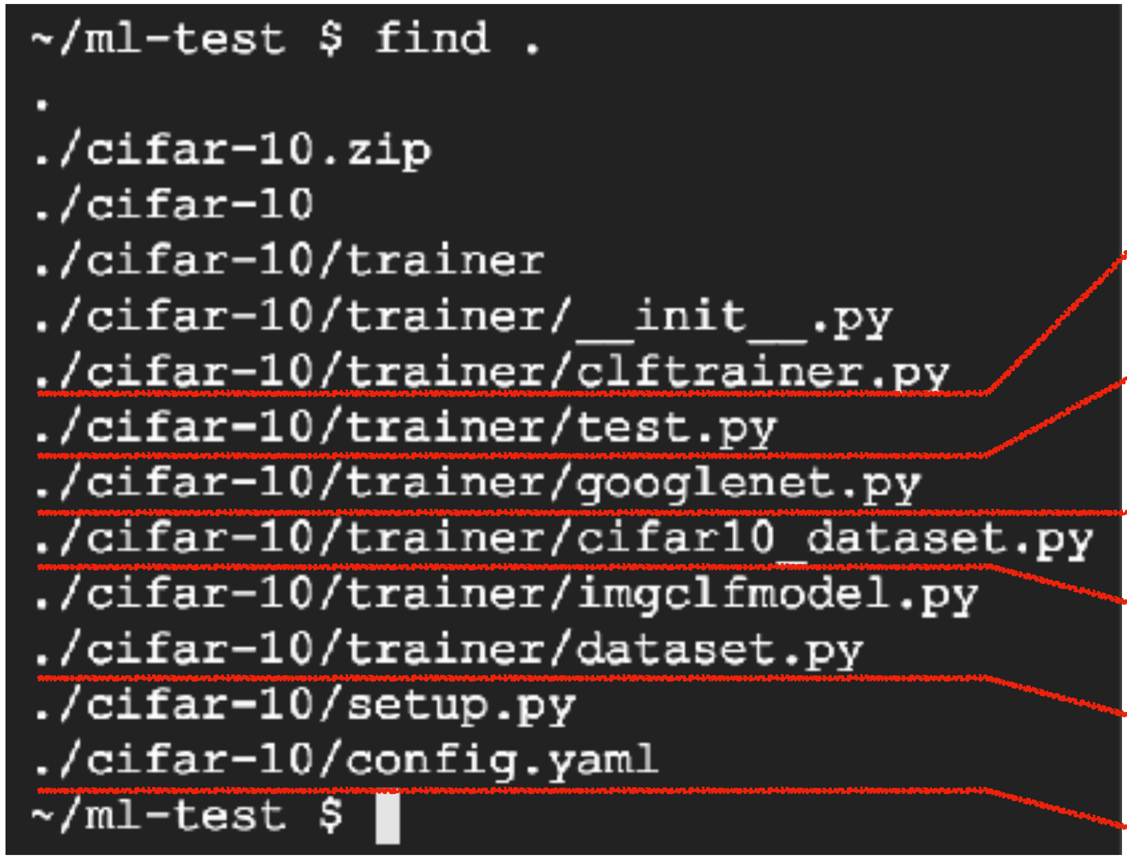


```
~ $ mkdir ml-test
~ $ mv cifar-10.zip ./ml-test
~ $ cd ml-test/
~/ml-test $ unzip cifar-10.zip
```

create a folder named “ml-test”

move cifar-10.zip file to the created folder

unzip the cifar-10.zip file after moving into the directory



```
~/ml-test $ find .
.
./cifar-10.zip
./cifar-10
./cifar-10/trainer
./cifar-10/trainer/__init__.py
./cifar-10/trainer/clftrainer.py
./cifar-10/trainer/test.py
./cifar-10/trainer/googlenet.py
./cifar-10/trainer/cifar10_dataset.py
./cifar-10/trainer/imgclfmodel.py
./cifar-10/trainer/dataset.py
./cifar-10/setup.py
./cifar-10/config.yaml
~/ml-test $ ▌
```

trainer class (can execute training)

where the main function is

GoogLeNet model definition

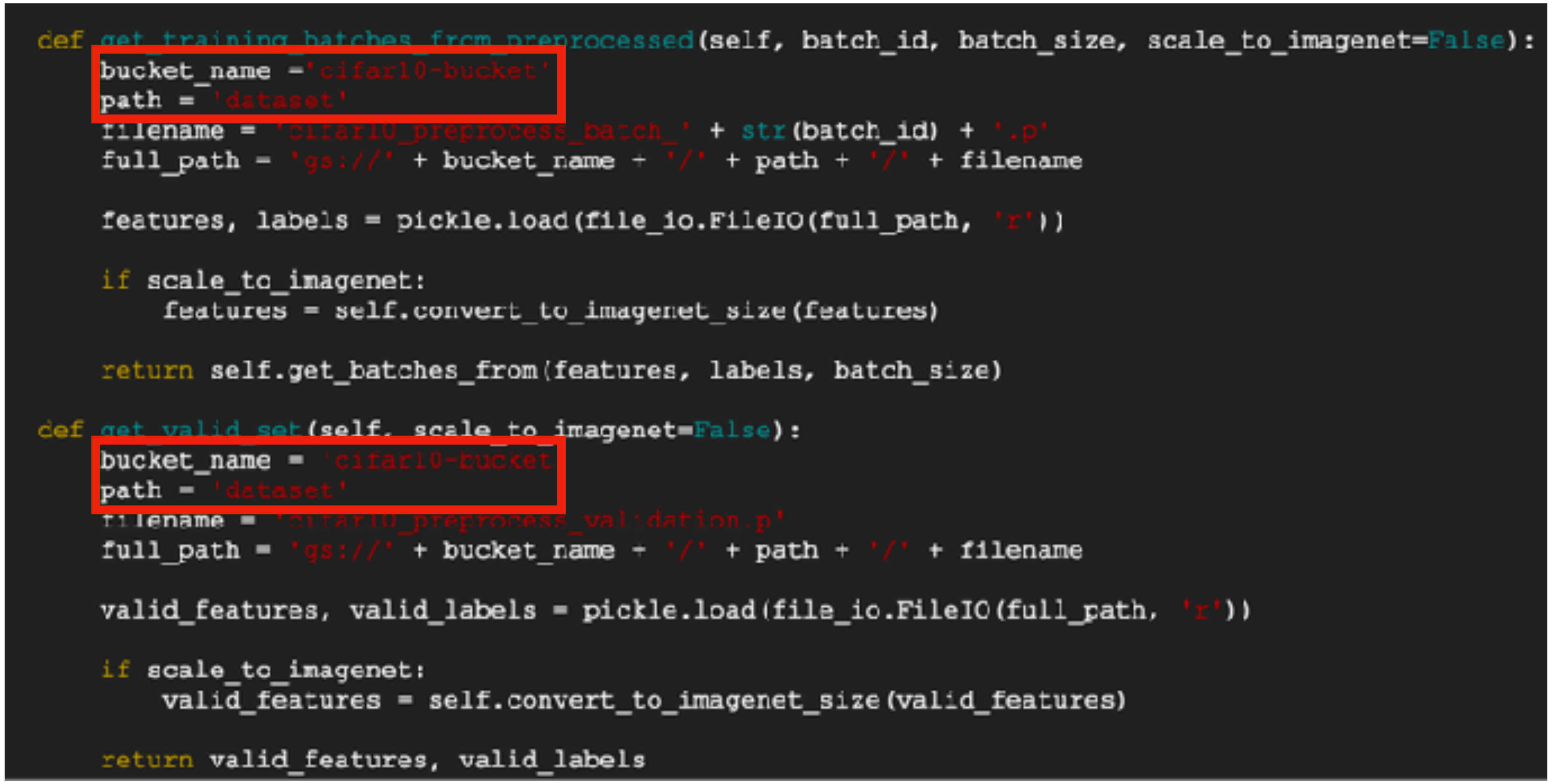
CIFAR10 dataset class (knows how to load)

base classes for model and dataset

config/setup files for cloud ml engine

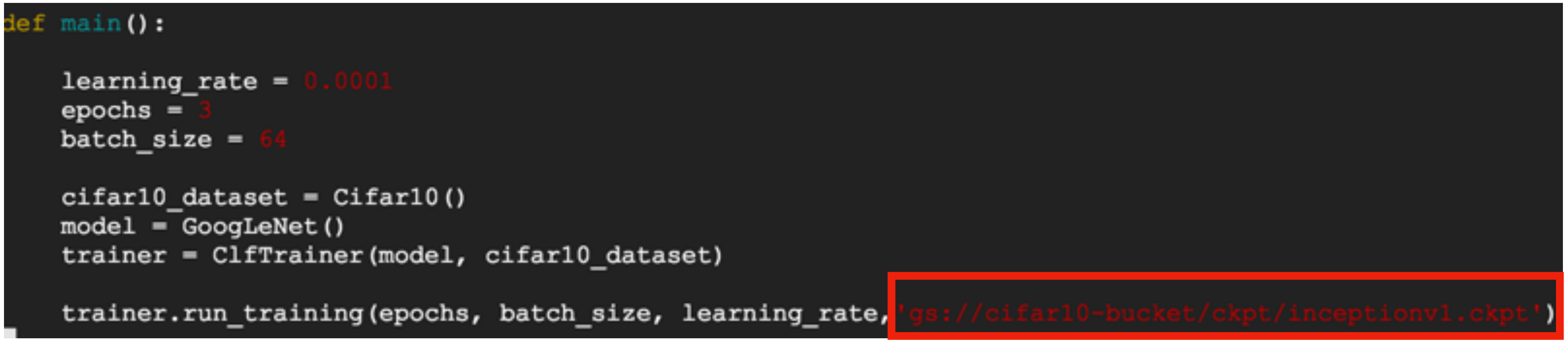
step5-set some variables

```
BUCKET_NAME=cifar10-bucket
REGION=us-central1

OUTPUT_PATH=gs://$BUCKET_NAME/output
JOB_NAME=hello_cifar10_v100
```



```
def get_training_batches_from_preprocessed(self, batch_id, batch_size, scale_to_imagenet=False):
    bucket_name ='cifar10-bucket'
    path = 'dataset'
    filename = 'cifar10_preprocess_batch_' + str(batch_id) + '.p'
    full_path = 'gs://' + bucket_name + '/' + path + '/' + filename

    features, labels = pickle.load(file_io.FileIO(full_path, 'r'))

    if scale_to_imagenet:
        features = self.convert_to_imagenet_size(features)

    return self.get_batches_from(features, labels, batch_size)

def get_valid_set(self, scale_to_imagenet=False):
    bucket_name = 'cifar10-bucket'
    path = 'dataset'
    filename = 'cifar10_preprocess_validation.p'
    full_path = 'gs://' + bucket_name + '/' + path + '/' + filename

    valid_features, valid_labels = pickle.load(file_io.FileIO(full_path, 'r'))

    if scale_to_imagenet:
        valid_features = self.convert_to_imagenet_size(valid_features)

    return valid_features, valid_labels
```

step5-set some variables(2)

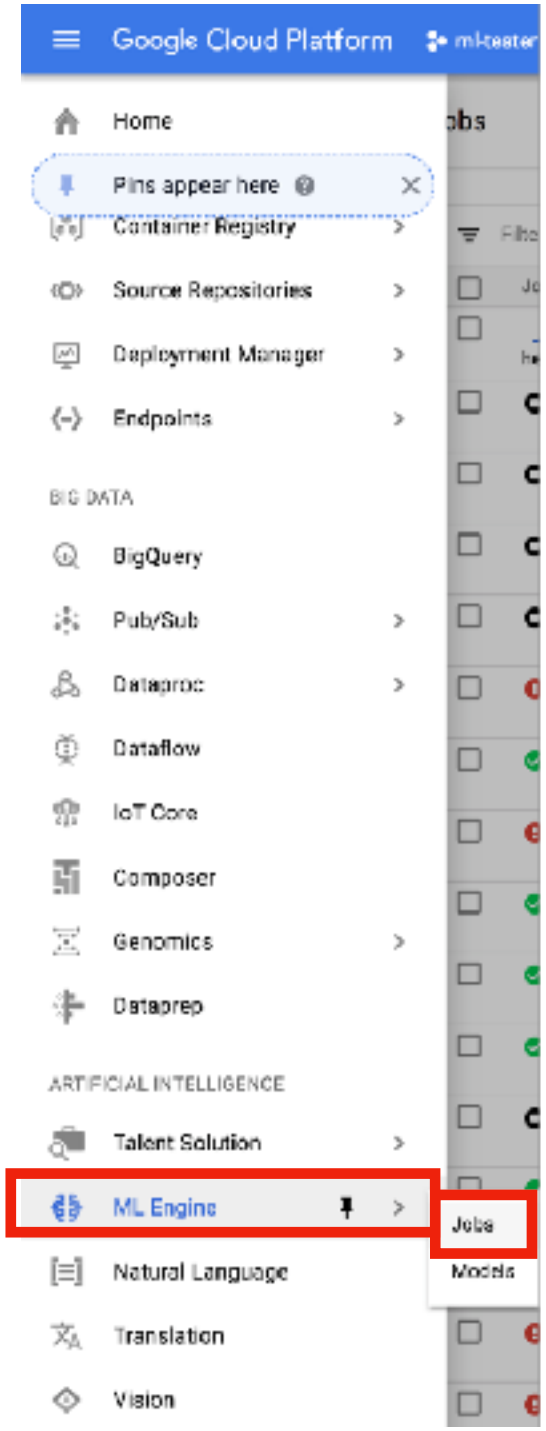


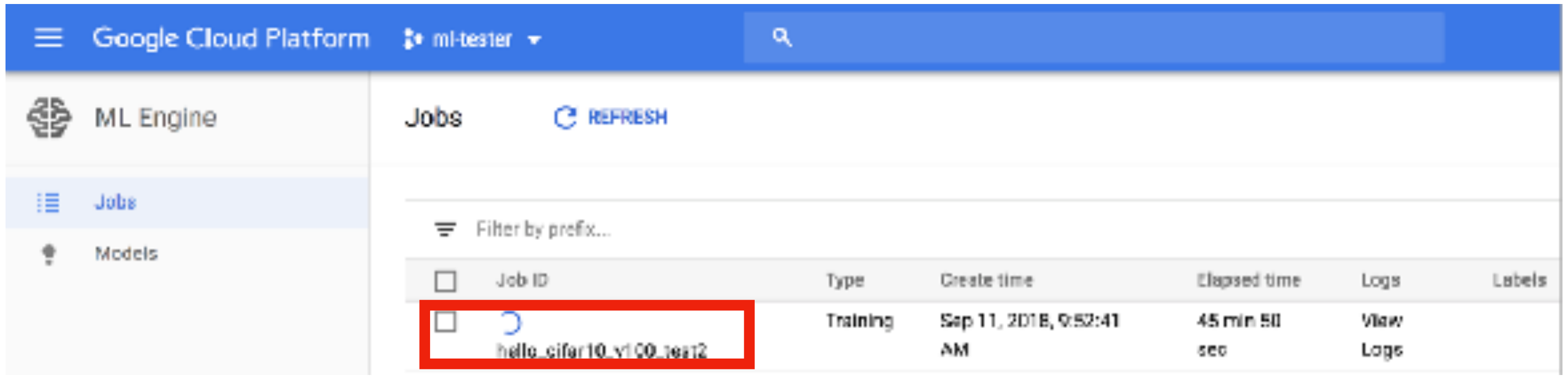
```python
def main():

    learning_rate = 0.0001
    epochs = 3
    batch_size = 64

    cifar10_dataset = Cifar10()
    model = GoogLeNet()
    trainer = ClfTrainer(model, cifar10_dataset)

    trainer.run_training(epochs, batch_size, learning_rate,'gs://cifar10-bucket/ckpt/inceptionv1.ckpt')
```

step6-run training

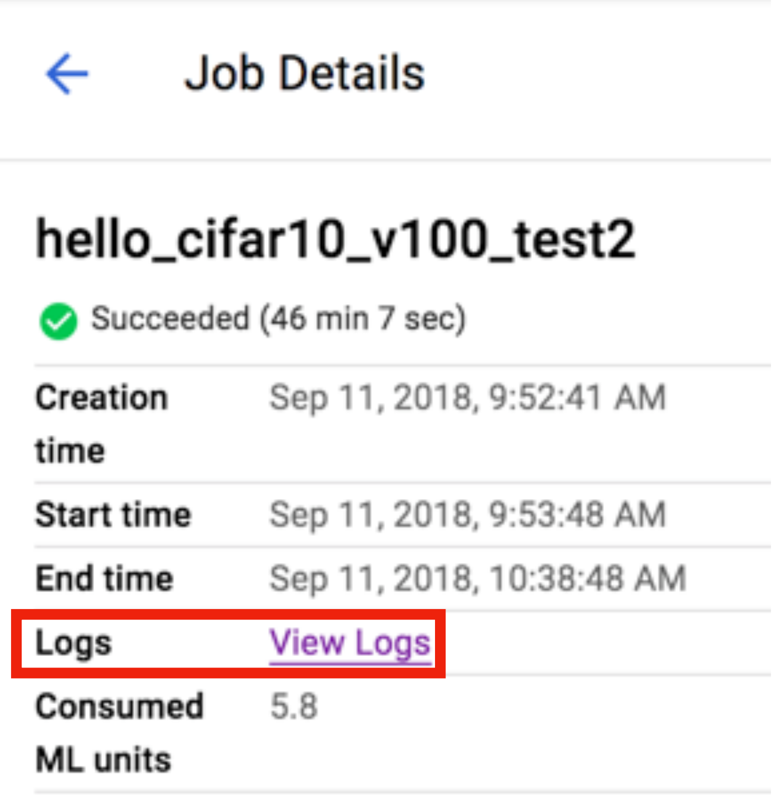
```
ml-test/cifar-10 $ gcloud ml-engine jobs submit training $JOB_NAME \
                   --package-path trainer \
                   --module-name trainer.test \
                   --job-dir $OUTPUT_PATH \
                   --region $REGION \
                   --config config.yaml \
                   --runtime-version 1.9
```

step7-watch/monitor





☐	Job ID	Type	Create time	Elapsed time	Logs	Labels
☐	hello_cifar10_v100_test2	Training	Sep 11, 2018, 9:52:41 AM	45 min 50 sec	View Logs	

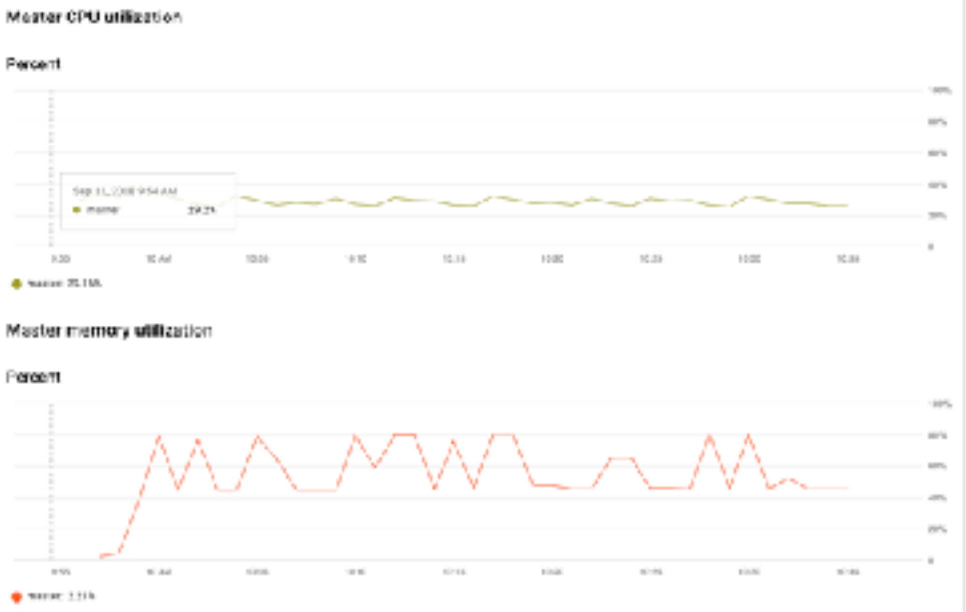


Job Details

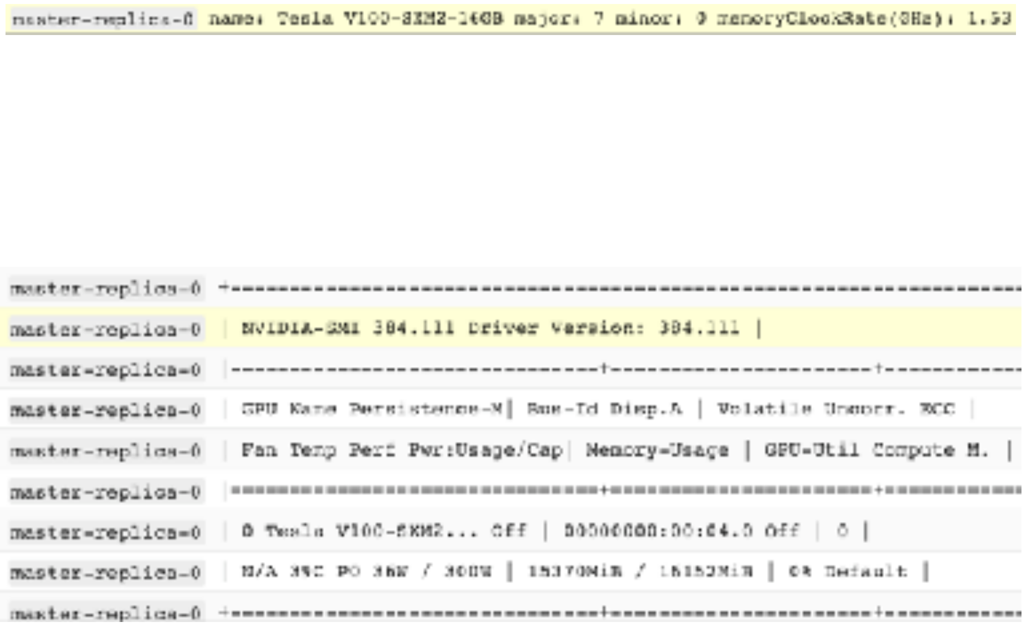
hello_cifar10_v100_test2

✓ Succeeded (46 min 7 sec)

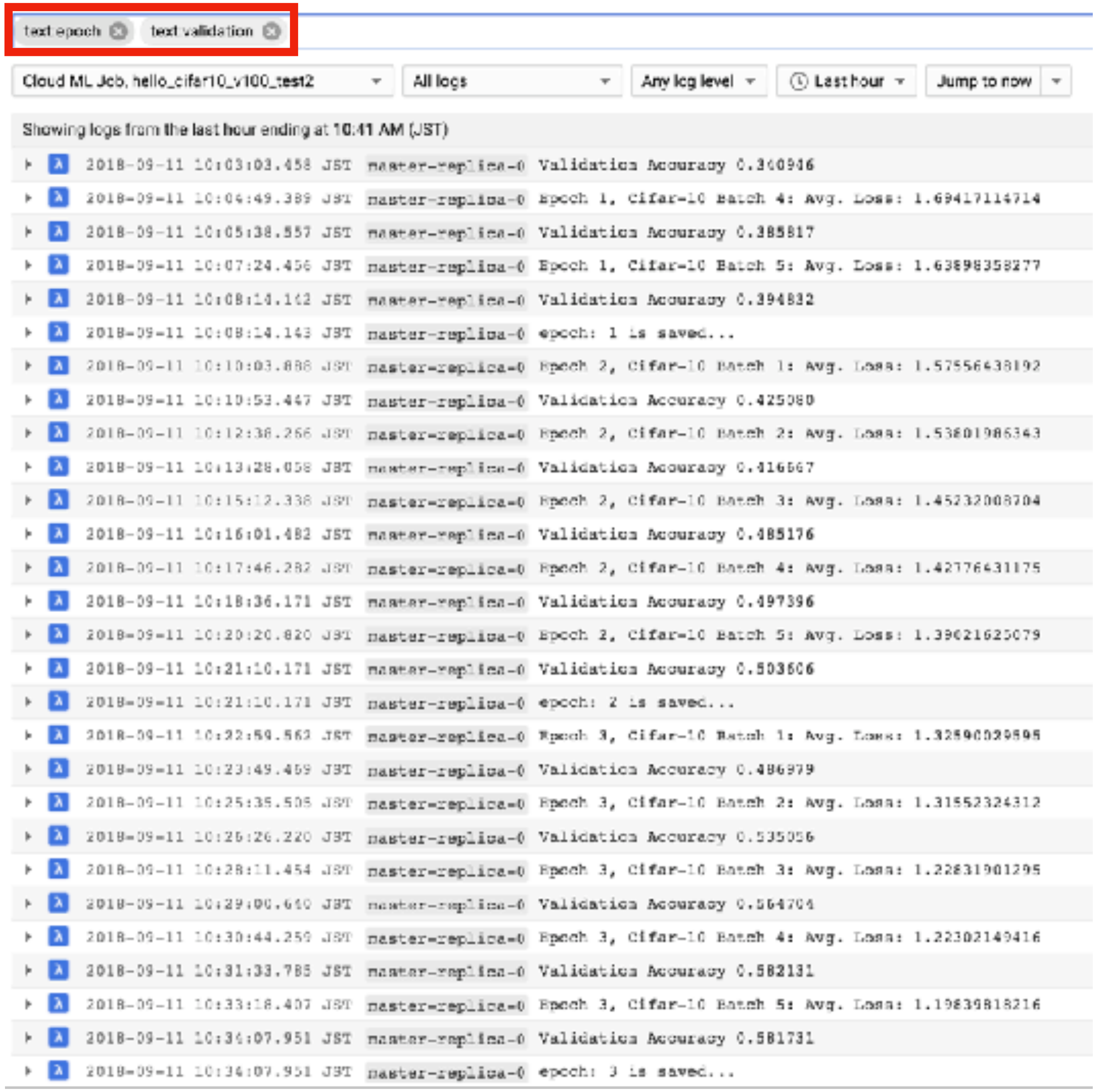
Creation time	Sep 11, 2018, 9:52:41 AM
Start time	Sep 11, 2018, 9:53:48 AM
End time	Sep 11, 2018, 10:38:48 AM
Logs	View Logs
Consumed ML units	5.8



step7-watch/monitor(2)



```
master-replica-0 name: Tesla V100-SXM2-16GB major: 7 minor: 0 memoryClockRate(GHz): 1.53

master-replica-0 +-----------------------------------------------------------------------------
master-replica-0 | NVIDIA-SMI 384.111 Driver Version: 384.111 |
master-replica-0 |-------------------------------+----------------------+----------------------
master-replica-0 | GPU Name Persistence-M| Bus-Id Disp.A | Volatile Uncorr. ECC |
master-replica-0 | Fan Temp Perf Pwr:Usage/Cap| Memory-Usage | GPU-Util Compute M. |
master-replica-0 |===============================+======================+======================
master-replica-0 | 0 Tesla V100-SXM2... Off | 00000000:00:04.0 Off | 0 |
master-replica-0 | N/A 35C P0 36W / 300W | 15370MiB / 16152MiB | 0% Default |
master-replica-0 +-------------------------------+----------------------+----------------------
```



text:epoch ⊗ text:validation ⊗

Cloud ML Job, hello_cifar10_v100_test2 ▾ | All logs ▾ | Any log level ▾ | Last hour ▾ | Jump to now ▾

Showing logs from the last hour ending at 10:41 AM (JST)

```
▸ 2018-09-11 10:03:03.458 JST master-replica-0 Validation Accuracy 0.360946
▸ 2018-09-11 10:04:49.389 JST master-replica-0 Epoch 1, Cifar-10 Batch 4: Avg. Loss: 1.69417114714
▸ 2018-09-11 10:05:38.557 JST master-replica-0 Validation Accuracy 0.385817
▸ 2018-09-11 10:07:24.456 JST master-replica-0 Epoch 1, Cifar-10 Batch 5: Avg. Loss: 1.63898358277
▸ 2018-09-11 10:08:14.142 JST master-replica-0 Validation Accuracy 0.394832
▸ 2018-09-11 10:08:14.143 JST master-replica-0 epoch: 1 is saved...
▸ 2018-09-11 10:10:03.888 JST master-replica-0 Epoch 2, Cifar-10 Batch 1: Avg. Loss: 1.57556438192
▸ 2018-09-11 10:10:53.447 JST master-replica-0 Validation Accuracy 0.425080
▸ 2018-09-11 10:12:38.266 JST master-replica-0 Epoch 2, Cifar-10 Batch 2: Avg. Loss: 1.53801986343
▸ 2018-09-11 10:13:28.058 JST master-replica-0 Validation Accuracy 0.416667
▸ 2018-09-11 10:15:12.338 JST master-replica-0 Epoch 2, Cifar-10 Batch 3: Avg. Loss: 1.45232008704
▸ 2018-09-11 10:16:01.482 JST master-replica-0 Validation Accuracy 0.485176
▸ 2018-09-11 10:17:46.282 JST master-replica-0 Epoch 2, Cifar-10 Batch 4: Avg. Loss: 1.42776431175
▸ 2018-09-11 10:18:36.171 JST master-replica-0 Validation Accuracy 0.497396
▸ 2018-09-11 10:20:20.820 JST master-replica-0 Epoch 2, Cifar-10 Batch 5: Avg. Loss: 1.39021625079
▸ 2018-09-11 10:21:10.171 JST master-replica-0 Validation Accuracy 0.503606
▸ 2018-09-11 10:21:10.171 JST master-replica-0 epoch: 2 is saved...
▸ 2018-09-11 10:22:59.563 JST master-replica-0 Epoch 3, Cifar-10 Batch 1: Avg. Loss: 1.32590029595
▸ 2018-09-11 10:23:49.459 JST master-replica-0 Validation Accuracy 0.486979
▸ 2018-09-11 10:25:35.505 JST master-replica-0 Epoch 3, Cifar-10 Batch 2: Avg. Loss: 1.31552324312
▸ 2018-09-11 10:26:26.220 JST master-replica-0 Validation Accuracy 0.535056
▸ 2018-09-11 10:28:11.454 JST master-replica-0 Epoch 3, Cifar-10 Batch 3: Avg. Loss: 1.22831901295
▸ 2018-09-11 10:29:00.640 JST master-replica-0 Validation Accuracy 0.564704
▸ 2018-09-11 10:30:44.259 JST master-replica-0 Epoch 3, Cifar-10 Batch 4: Avg. Loss: 1.22302149416
▸ 2018-09-11 10:31:33.785 JST master-replica-0 Validation Accuracy 0.582131
▸ 2018-09-11 10:33:18.407 JST master-replica-0 Epoch 3, Cifar-10 Batch 5: Avg. Loss: 1.19839818216
▸ 2018-09-11 10:34:07.951 JST master-replica-0 Validation Accuracy 0.581731
▸ 2018-09-11 10:34:07.951 JST master-replica-0 epoch: 3 is saved...
```

appendix

- how to get current project ID?

  ```
  $(gcloud config list project --format “value(core.project)”)
  ```

-
  ```
  with tf.device(‘/device:GPU:0’):
  ```

Google
Cloud Platform

Cloud
ML Engine