**Google Cloud Platform Setup**

1. **Sign up for a Google Cloud Platform free trial account**

[Sign up](https://cloud.google.com/ml-engine/) with 12-months free trial and $300 free trial credits. Account type should be “individual”. You need to add a payment method (credit card/ debit card/ bank account) for verification purpose.

1. **Set up your Google Cloud project**

[Create a new Cloud Platform project](https://console.cloud.google.com/project?_ga=1.197152431.1429848579.1482169659). This is where your project lives. Click Create Project and follow instructions.

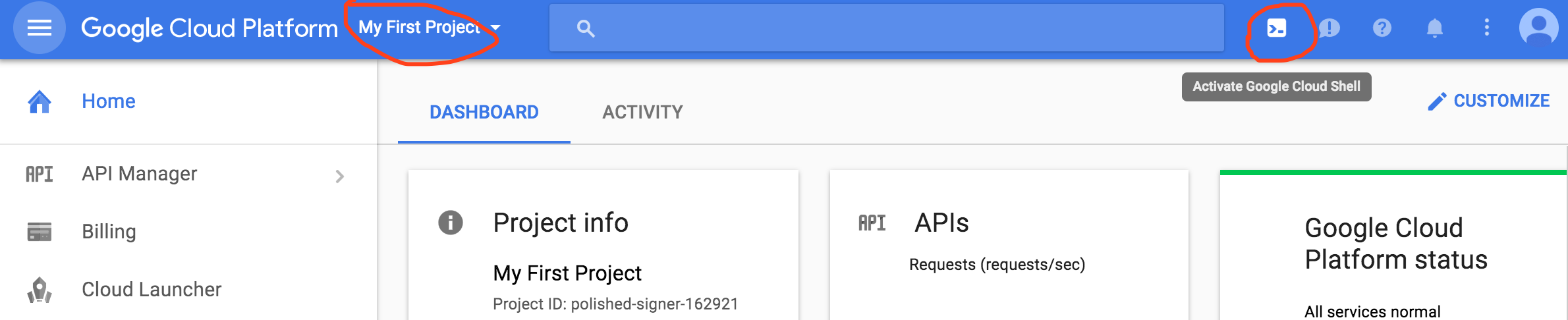
1. [**Enable the APIs**](https://console.cloud.google.com/flows/enableapi?apiid=ml.googleapis.com,dataflow,compute_component,logging,storage_component,storage_api,bigquery&_ga=1.205652499.1429848579.1482169659)**but ignore adding Credentials.**

Click the link, and this enables the set of Cloud APIs that are needed for Cloud ML functionality such as Cloud Logging to get your training logs. Other APIs include: Cloud Machine Learning, Dataflow, Compute Engine, Cloud Storage, Cloud Storage JSON, and BigQuery APIs.

You may have to select the newly-created project. Expect this to take some time. After the APIs are enabled, do not “Go to Credentials”

1. **Set up your environment using cloud shell**

Select your newly-created project at the top of the window besides “Google Cloud Platform”, then click “Activate Google Cloud Shell” button on the top right. On the window pop up, click “START CLOUD SHELL”.



Then run the following commands in the shell:

The first step to setting up the environment is to configure the gcloud command-line tool to use your selected project. For more information on the pre-installed packages in Cloud ML, refer to [this thread](http://stackoverflow.com/questions/41938862/what-is-the-default-environment-on-google-cloud-ml-vms).

gcloud config set project [selected-project-id]

(My [selected-project-id] as shown in the above screenshot, is polished-signer-162921)

Install the latest version of TensorFlow (1.0, RC2) with the following 2 command lines.

pip download tensorflow

pip install --user -U tensorflow\*.whl

If you got something like

99% |████████████████████████████████| 44.1MB 49.5MB/s eta 0:00:01Killed

while downloading tensorflow, try this instead of the above two commands:

pip install --user -U tensorflow

To clear the console, simply type “clear”.

1. **Verify the Google Cloud SDK Components**

SDK – software development kit

List the models to verify that the command returns an empty list.

gcloud ml-engine models list

The command will an empty list, and after you start creating models, you can see them listed by using this command.

Listed 0 items.

1. **Connect to and Run the Starter Code**

Starter code is provided in github, and you can run in Cloud ML as a starting point to learn how to train, evaluate, and create predictions.

In your cloud shell, type the following into the command line to clone the [YouTube-8M github repo](https://github.com/google/youtube-8m.git):

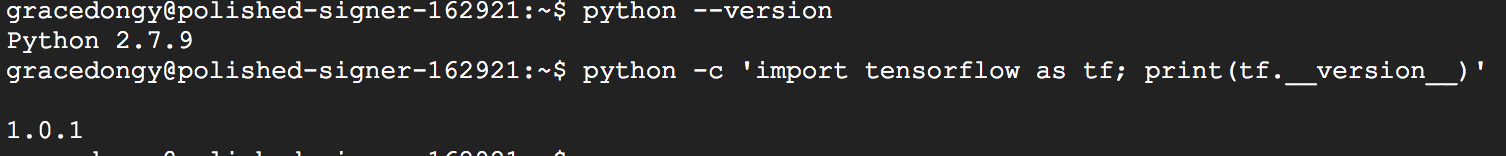
git clone https://github.com/google/youtube-8m.git

The following starter code comes from the github repo [README](https://github.com/google/youtube-8m/blob/master/README.md). The code gives an end-to-end working example for reading the dataset, training a TensorFlow model, and evaluating the performance of the model. Out of the box, you can train several [model architectures](https://github.com/google/youtube-8m/blob/master/README.md#overview-of-models) over either frame-level or video-level features. The code can easily be extended to train your own custom-defined models.

1. Please verify that you have Python 2.7+ and Tensorflow 1.0.0 or higher installed by running the following commands:

python --version

python -c 'import tensorflow as tf; print(tf.\_\_version\_\_)'



1. Training on the Cloud over Video-Level Features

BUCKET\_NAME=gs://${USER}\_yt8m\_train\_bucket

# (One Time) Create a storage bucket to store training logs and checkpoints.

gsutil mb -l us-east1 $BUCKET\_NAME

# Submit the training job.

JOB\_NAME=yt8m\_train\_$(date +%Y%m%d\_%H%M%S); gcloud --verbosity=debug ml-engine jobs \

submit training $JOB\_NAME \

--package-path=youtube-8m --module-name=youtube-8m.train \

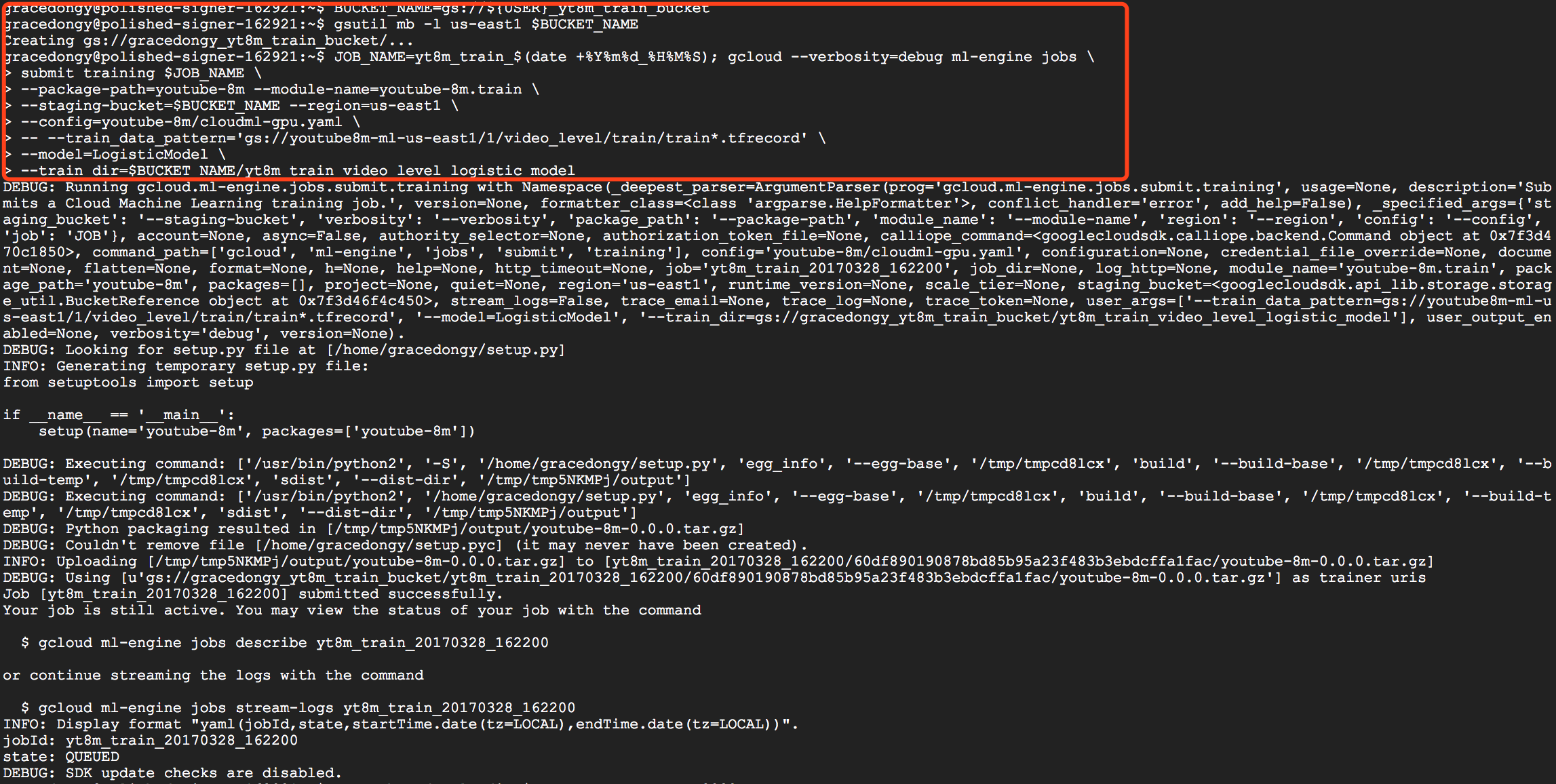
--staging-bucket=$BUCKET\_NAME --region=us-east1 \

--config=youtube-8m/cloudml-gpu.yaml \

-- --train\_data\_pattern='gs://youtube8m-ml-us-east1/1/video\_level/train/train\*.tfrecord' \

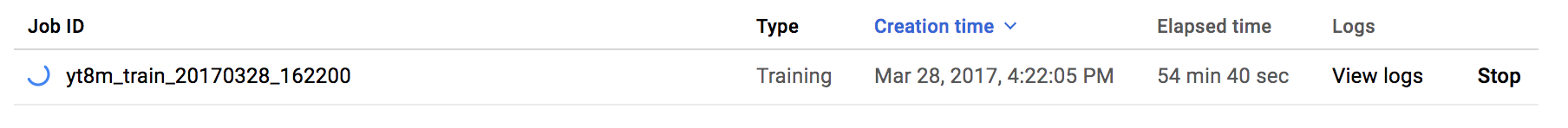
--model=LogisticModel \

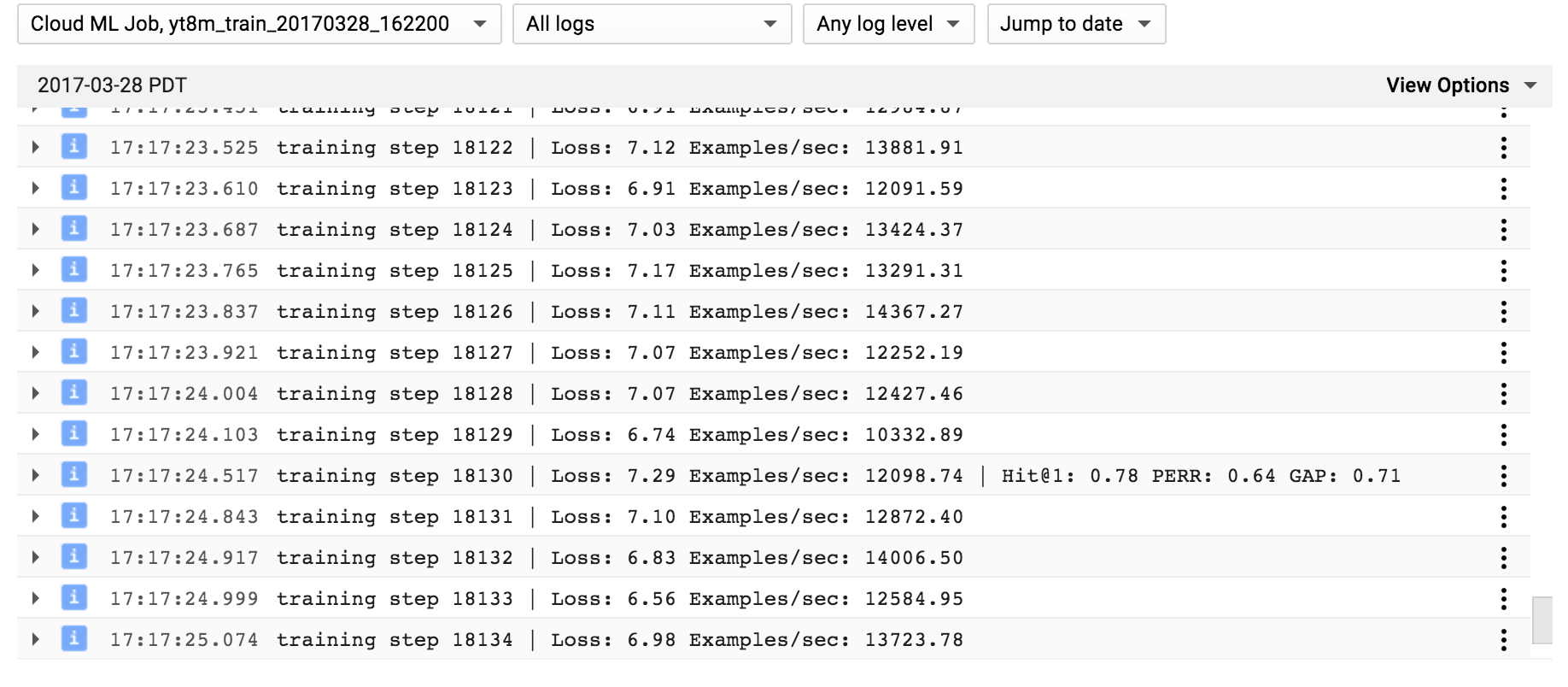
--train\_dir=$BUCKET\_NAME/yt8m\_train\_video\_level\_logistic\_model



In the 'gsutil' command above, the 'package-path' flag refers to the directory containing the [train.py](https://github.com/google/youtube-8m/blob/master/train.py) script and more generally the python package which should be deployed to the cloud worker. The module-name refers to the specific python script which should be executed (in this case the train module, the train.py file).

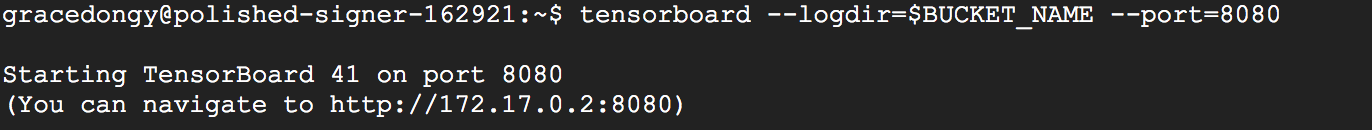
It may take several minutes before the job starts running on Google Cloud (about 7 mins according to my experience). At this point you can disconnect your console by pressing "ctrl-c". You can check on its progress or halt the job by visiting the [Google Cloud ML Jobs console](https://console.cloud.google.com/ml/jobs) and will see outputs like the following:



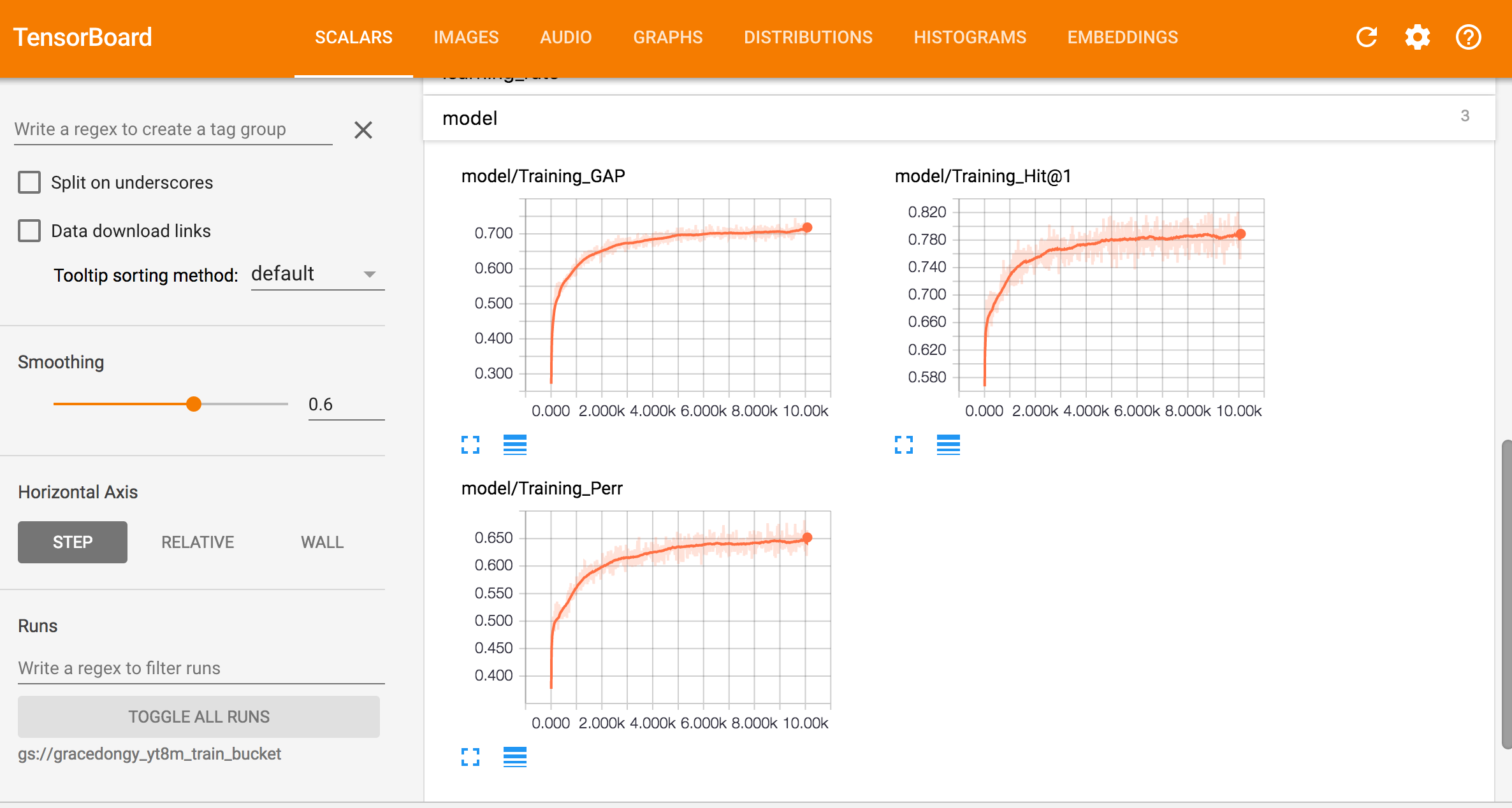


You can train many jobs at once and use tensorboard to compare their performance visually.

tensorboard --logdir=$BUCKET\_NAME --port=8080



Once tensorboard is running, you can click the Web Preview button on the upper left corner of the Cloud Shell window and select "Preview on port 8080". This will bring up a new browser tab with the Tensorboard view.



**Jobs do not automatically stop.**

For the actual competition, you should monitor your training jobs and stop them manually when the model has converged (e.g. based on the TensorBoard graphs).

To stop a job: click “Stop” button in the Jobs Console.

1. Using Frame-Level Features

Append

--frame\_features=True --model=FrameLevelLogisticModel --feature\_names="rgb" \

--feature\_sizes="1024" --batch\_size=128 \

--train\_dir=$BUCKET\_NAME/yt8m\_train\_frame\_level\_logistic\_model

to the 'gcloud' commands given above, and change 'video\_level' in paths to 'frame\_level'. Here is a sample command to kick-off a frame-level job:

JOB\_NAME=yt8m\_train\_$(date +%Y%m%d\_%H%M%S); gcloud --verbosity=debug ml-engine jobs \

submit training $JOB\_NAME \

--package-path=youtube-8m --module-name=youtube-8m.train \

--staging-bucket=$BUCKET\_NAME --region=us-east1 \

--config=youtube-8m/cloudml-gpu.yaml \

-- --train\_data\_pattern='gs://youtube8m-ml-us-east1/1/frame\_level/train/train\*.tfrecord' \

--frame\_features=True --model=FrameLevelLogisticModel --feature\_names="rgb" \

--feature\_sizes="1024" --batch\_size=128 \

--train\_dir=$BUCKET\_NAME/yt8m\_train\_frame\_level\_logistic\_model

The 'FrameLevelLogisticModel' is designed to provide equivalent results to a logistic model trained over the video-level features. Please look at the ['video\_level\_models.py'](https://github.com/google/youtube-8m/blob/master/video_level_models.py) or ['frame\_level\_models.py'](https://github.com/google/youtube-8m/blob/master/frame_level_models.py) files to see how to implement your own models.

1. Using Audio Features

The feature files (both Frame-Level and Video-Level) contain two sets of features: 1) visual and 2) audio. The code defaults to using the visual features only, but it is possible to use audio features instead of (or besides) visual features. To specify the (combination of) features to use you must set --feature\_names and --feature\_sizes flags. The visual and audio features are called 'rgb' and 'audio' and have 1024 and 128 dimensions, respectively. The two flags take a comma-separated list of values in string. For example, to use audio-visual Video-Level features the flags must be set as follows:

--feature\_names="mean\_rgb, mean\_audio" --feature\_sizes="1024, 128"

Similarly, to use audio-visual Frame-Level features use:

--feature\_names="rgb, audio" --feature\_sizes="1024, 128"

**NOTE:** Make sure the set of features and the order in which the appear in the lists provided to the two flags above match. Also, the order must match when running training, evaluation, or inference.

1. Evaluation and Inference

Here's how to evaluate a model on the validation dataset:

JOB\_TO\_EVAL= yt8m\_train\_video\_level\_logistic\_model

JOB\_NAME=yt8m\_eval\_$(date +%Y%m%d\_%H%M%S); gcloud --verbosity=debug ml-engine jobs \

submit training $JOB\_NAME \

--package-path=youtube-8m --module-name=youtube-8m.eval \

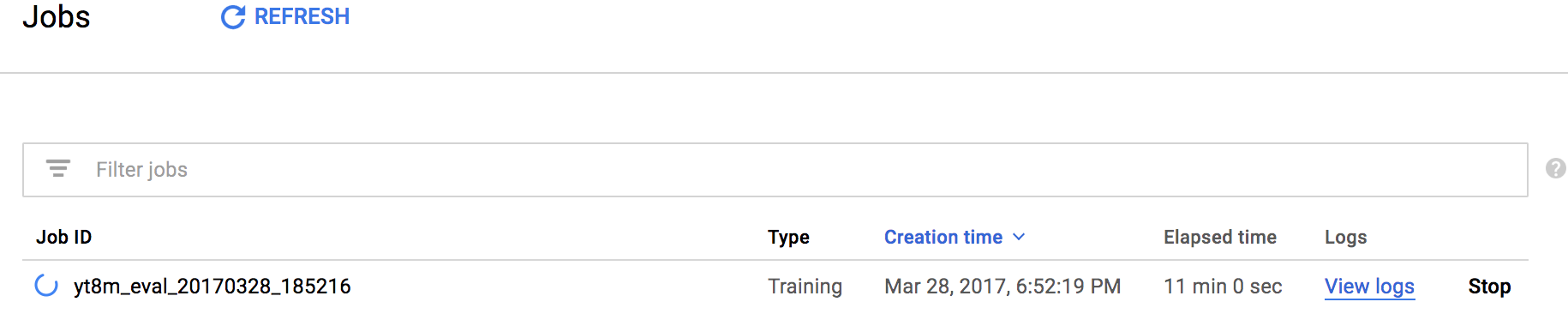
--staging-bucket=$BUCKET\_NAME --region=us-east1 \

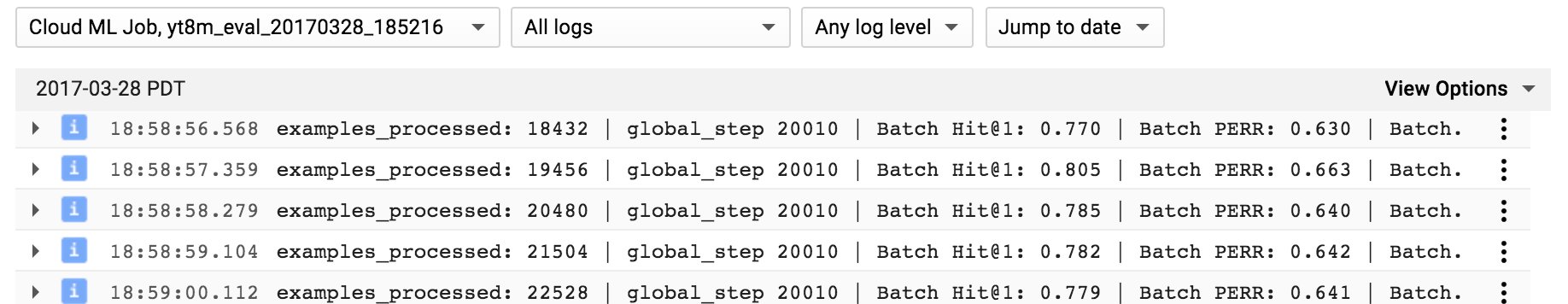
--config=youtube-8m/cloudml-gpu.yaml \

-- --eval\_data\_pattern='gs://youtube8m-ml-us-east1/1/video\_level/validate/validate\*.tfrecord' \

--model=LogisticModel \

--train\_dir=$BUCKET\_NAME/${JOB\_TO\_EVAL} --run\_once=True







The evaluation process uses the [eval.py](https://github.com/google/youtube-8m/blob/master/eval.py) script.

And here's how to perform inference with a model on the test set:

JOB\_TO\_EVAL= yt8m\_train\_video\_level\_logistic\_model

JOB\_NAME=yt8m\_inference\_$(date +%Y%m%d\_%H%M%S); gcloud --verbosity=debug ml-engine jobs \

submit training $JOB\_NAME \

--package-path=youtube-8m --module-name=youtube-8m.inference \

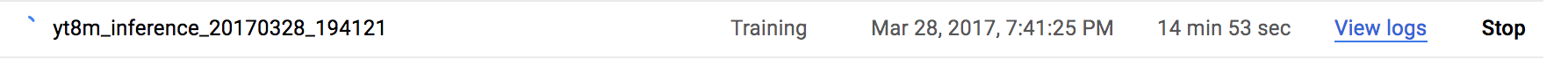
--staging-bucket=$BUCKET\_NAME --region=us-east1 \

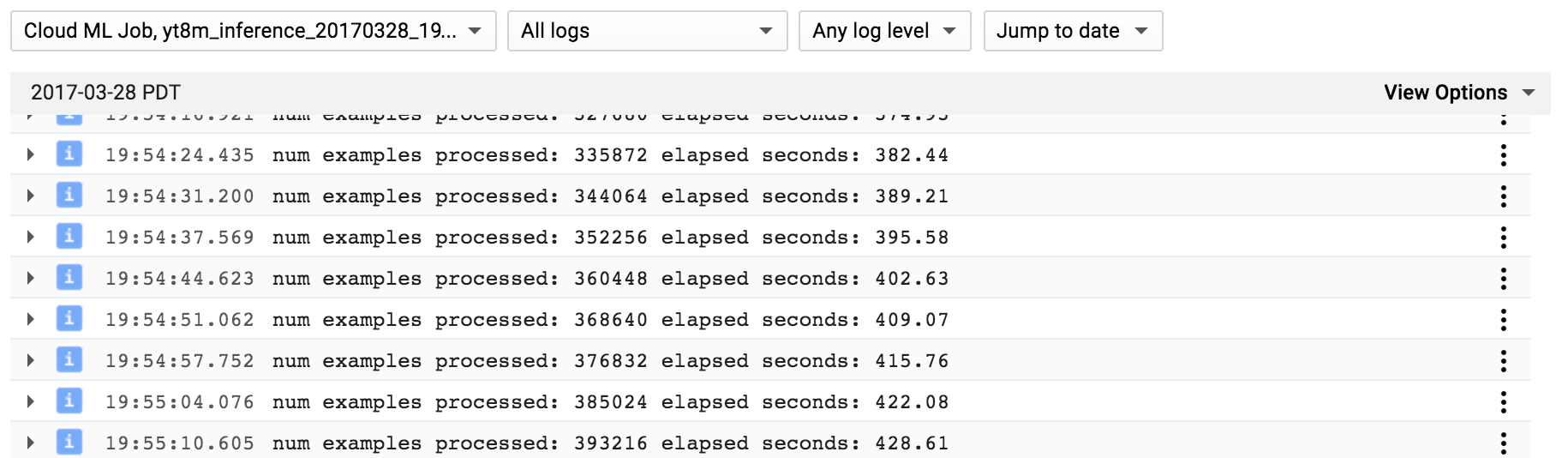
--config=youtube-8m/cloudml-gpu.yaml \

-- --input\_data\_pattern='gs://youtube8m-ml/1/video\_level/test/test\*.tfrecord' \

--train\_dir=$BUCKET\_NAME/${JOB\_TO\_EVAL} \

--output\_file=$BUCKET\_NAME/${JOB\_TO\_EVAL}/predictions.csv







Note the confusing use of 'training' in the above gcloud commands. Despite the name, the 'training' argument really just offers a cloud hosted python/tensorflow service. From the point of view of the Cloud Platform, there is no distinction between our training and inference jobs. The Cloud ML platform also offers specialized functionality for prediction with Tensorflow models, but discussing that is beyond the scope of this readme.

**Inference Using Batch Prediction**

To perform inference faster, you can also use the Cloud ML batch prediction service.

First, find the directory where the training job exported the model:

gsutil list ${BUCKET\_NAME}/yt8m\_train\_video\_level\_logistic\_model/export

You should see an output similar to this one:

${BUCKET\_NAME}/yt8m\_train\_video\_level\_logistic\_model/export/

${BUCKET\_NAME}/yt8m\_train\_video\_level\_logistic\_model/export/step\_1/

${BUCKET\_NAME}/yt8m\_train\_video\_level\_logistic\_model/export/step\_1001/

${BUCKET\_NAME}/yt8m\_train\_video\_level\_logistic\_model/export/step\_2001/

${BUCKET\_NAME}/yt8m\_train\_video\_level\_logistic\_model/export/step\_3001/

Select the latest version of the model that was saved. For instance, in our case, we select the version of the model that was saved at step 3001:

EXPORTED\_MODEL\_DIR=${BUCKET\_NAME}/yt8m\_train\_video\_level\_logistic\_model/export/step\_3001/

Start the batch prediction job using the following command:

JOB\_NAME=yt8m\_batch\_predict\_$(date +%Y%m%d\_%H%M%S); \

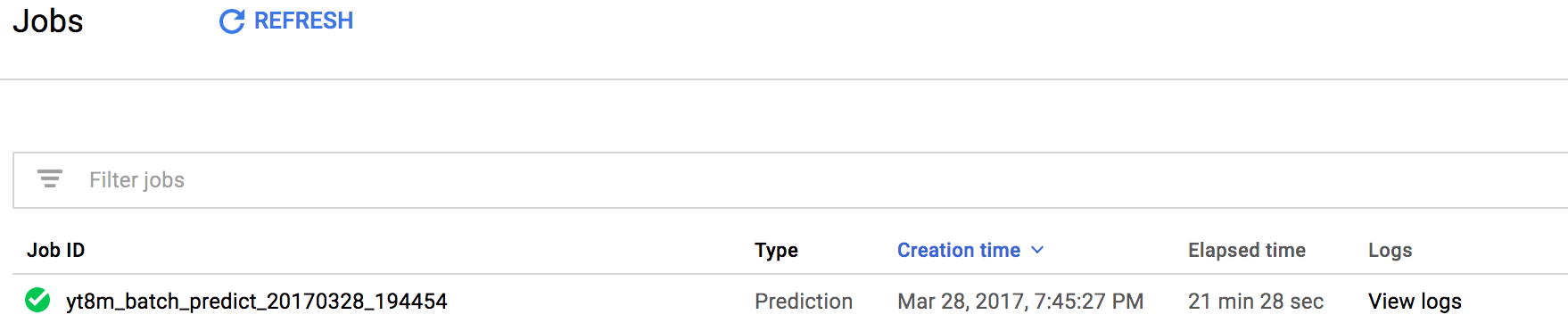
gcloud ml-engine jobs submit prediction ${JOB\_NAME} --verbosity=debug \

--model-dir=${EXPORTED\_MODEL\_DIR} --data-format=TF\_RECORD \

--input-paths=gs://youtube8m-ml/1/video\_level/test/test\* \

--output-path=${BUCKET\_NAME}/batch\_predict/${JOB\_NAME} --region=us-east1 \

--runtime-version=1.0 --max-worker-count=10



You can check the progress of the job on the [Google Cloud ML Jobs console](https://console.cloud.google.com/ml/jobs). To have the job complete faster, you can increase 'max-worker-count' to a higher value.

Once the batch prediction job has completed, turn its output into a submission in the CSV format by running the following commands:

# Copy the output of the batch prediction job to a local directory

mkdir -p /tmp/batch\_predict/${JOB\_NAME}

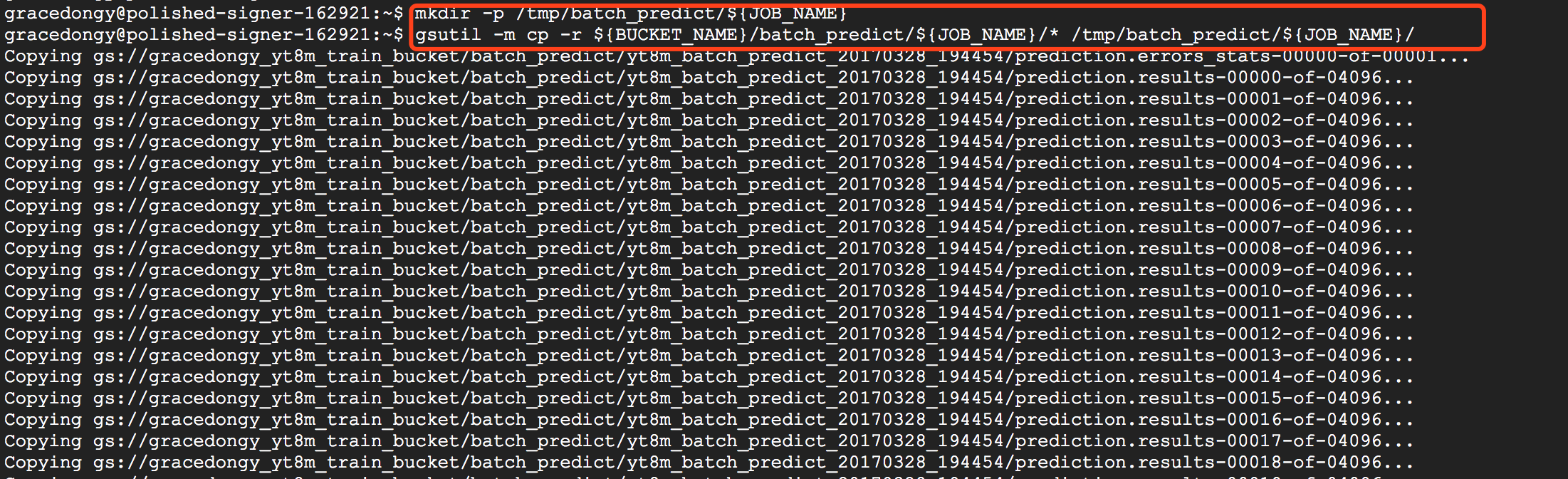
gsutil -m cp -r ${BUCKET\_NAME}/batch\_predict/${JOB\_NAME}/\* /tmp/batch\_predict/${JOB\_NAME}/

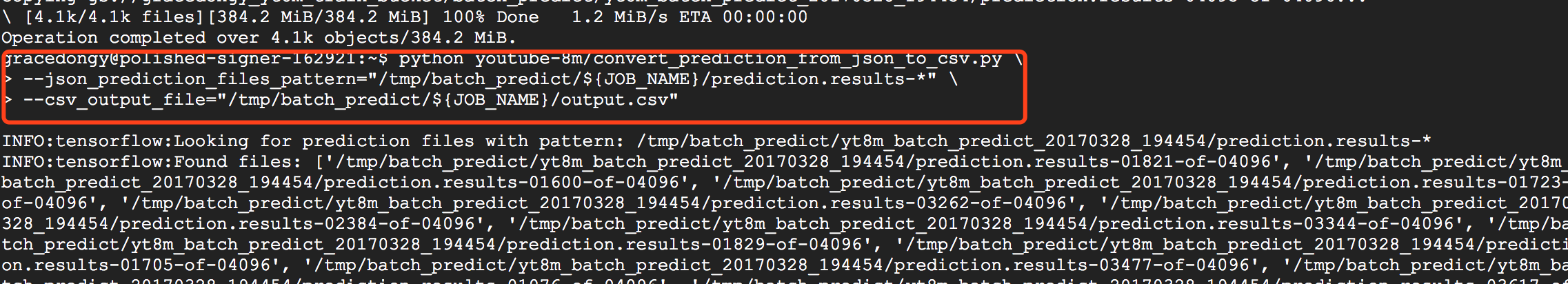
# Convert the output of the batch prediction job into a CSV file ready for submission

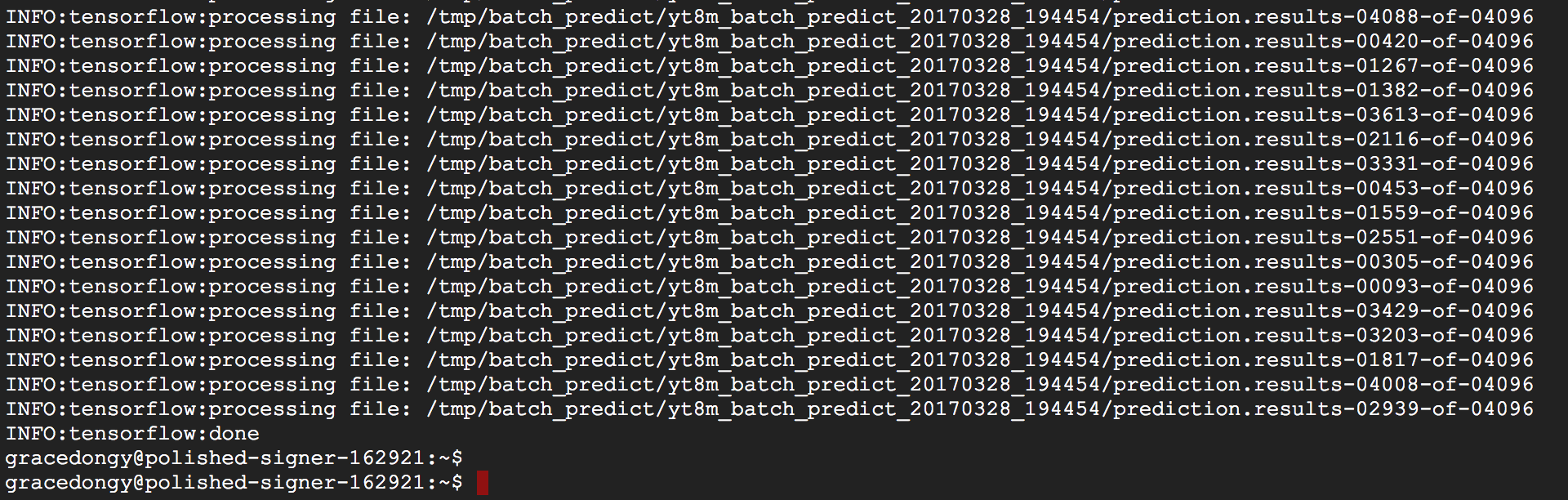
python youtube-8m/convert\_prediction\_from\_json\_to\_csv.py \

--json\_prediction\_files\_pattern="/tmp/batch\_predict/${JOB\_NAME}/prediction.results-\*" \

--csv\_output\_file="/tmp/batch\_predict/${JOB\_NAME}/output.csv"





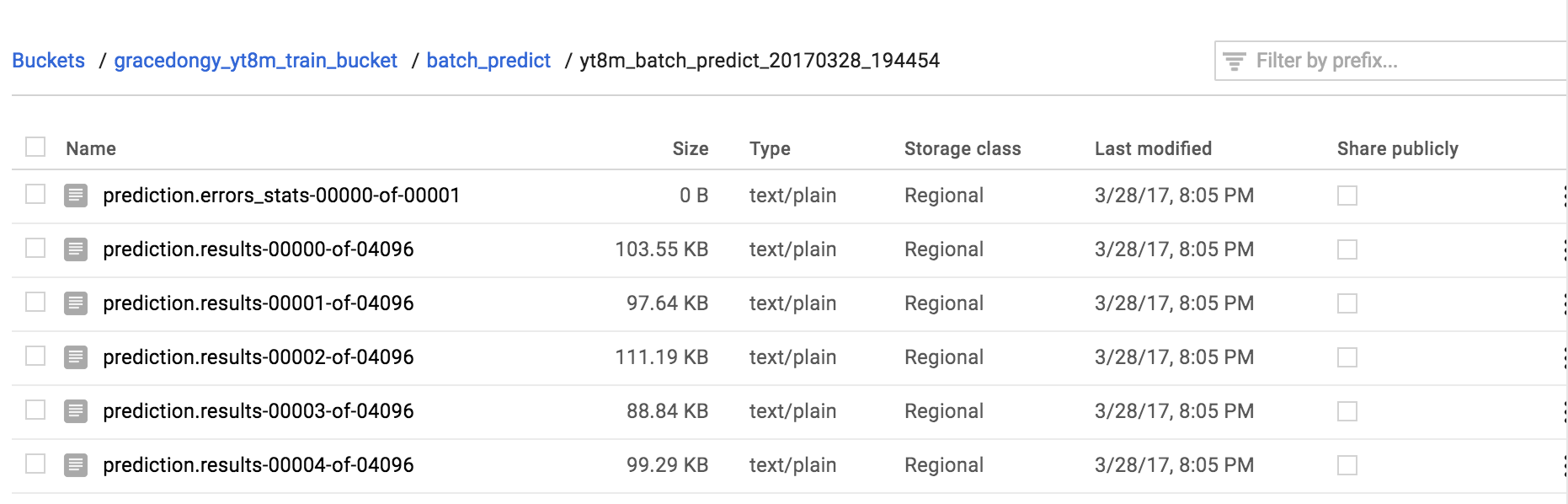


1. Accessing Files on Google Cloud

You can browse the storage buckets you created on Google Cloud, for example, to access the trained models, prediction CSV files, etc. by visiting the [Google Cloud storage browser](https://console.cloud.google.com/storage/browser).

Alternatively, you can use the 'gsutil' command to download the files directly. For example, to download the output of the inference code from the previous section to your local machine, run:

gsutil cp $BUCKET\_NAME/${JOB\_TO\_EVAL}/predictions.csv .



1. **Download your Predictions File to make a Kaggle Submission**

Running the inference command above generates a predictions file (predictions.csv). A gs:// link should be given to you when your inference job is finished. You can find this at the end of the log. Download your predictions.csv file to your local computer as instructed above.

