CITS5551 &CITS5552 Software Engineering Design Project

Weed Detection - Maintenance Manual

Jiaqi Han 22292229

I. Introduction

This document describes maintenance process of the project and is intended as a guide for future developers to reproduce and alter the processes by which the final product is made live. It mainly focuses on how to deploy the Mask_RCNN on Google Cloud Platform. If you have any other questions, you can send email to Google Support Team.

II. Deployment - Google Cloud Platform

The Google cloud platform can provide GPU and huge storage without PC limitations.

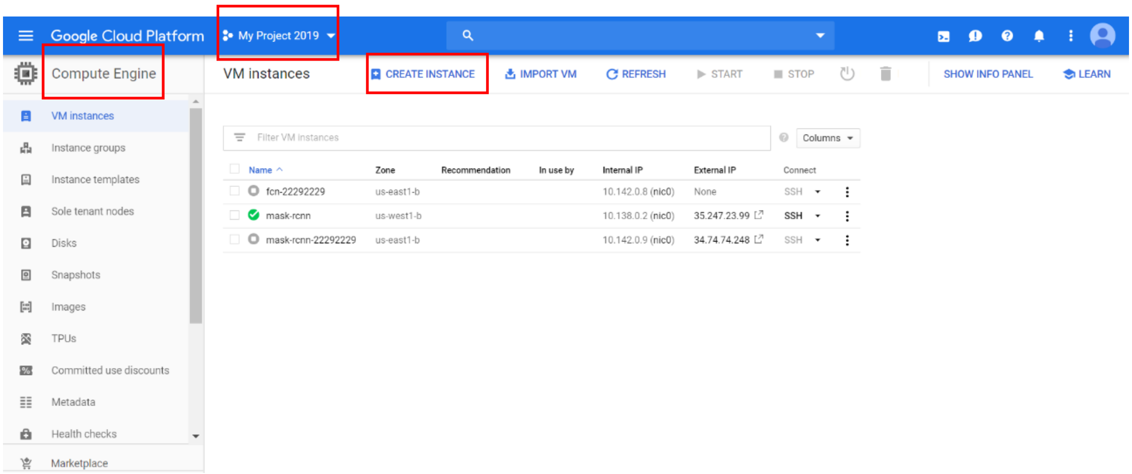


Figure 1: 'mask-rcnn' Weed Detection VM instances of Google Cloud platform

Steps in detail:

Step 1: Apply GPU quotas: The algorithm will run faster and better with GPU support, so the first step is to apply GPU quotas for the VM instance of Weed Detection project. For the project "My Project 2019" I have applied 4 quotas for Global and 4 quotas for GPU "NVIDIA P100 GPUs"

Choose menu: IAM & admin/Quotas → Setting parameters: Services as "Compute Engine API" / Metric as "NVIDIA P100 GPUs" (the kind of GPU you want)/ Region as "us-east1" & "Global" (Not all regions have GPUs support) → Edit Quotas: give your personal details and reasons, the email will be sent to Google Support Team automatically.

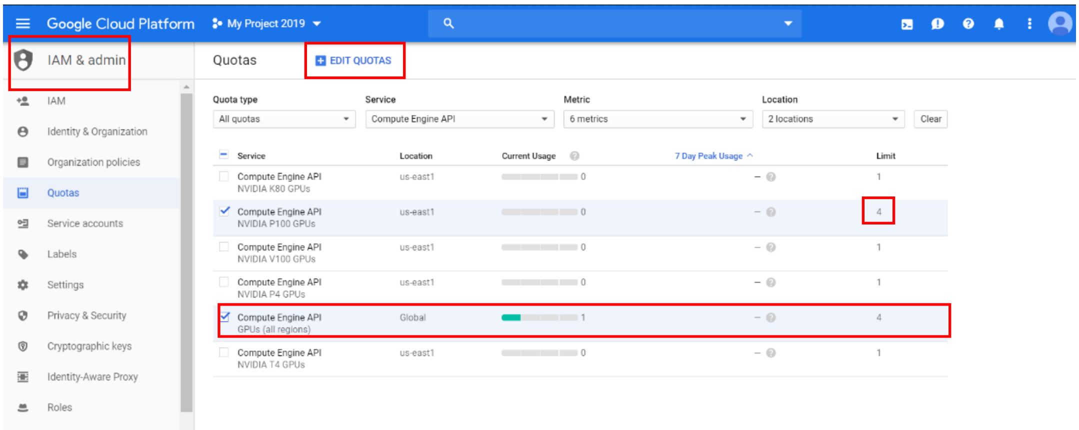
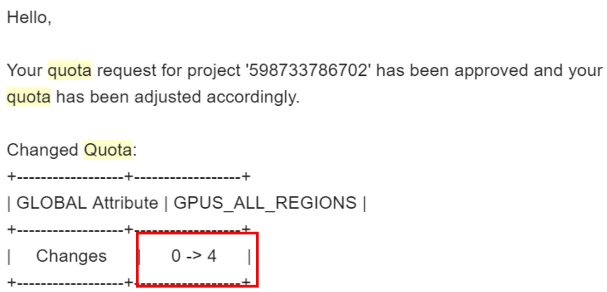


Figure 2: GPU quotas checking and application for the project

The application is successful once received such emails. (Sometimes they may forget to increase the Global Quota, best to apply both quotas at the same time.)



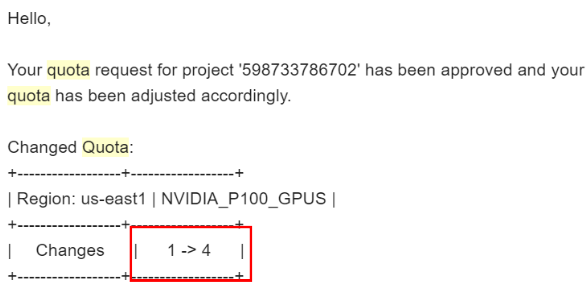
Hello,

Your quota request for project '598733786702' has been approved and your quota has been adjusted accordingly.

Changed Quota:

```
+------------------+------------------+
| GLOBAL Attribute | GPUS_ALL_REGIONS |
+------------------+------------------+
|    Changes       |      0 -> 4      |
+------------------+------------------+
```

Figure 3: Global GPU quotas confirmation email



Hello,

Your quota request for project '598733786702' has been approved and your quota has been adjusted accordingly.

Changed Quota:

```
+------------------+------------------+
| Region: us-east1 | NVIDIA_P100_GPUS |
+------------------+------------------+
|    Changes       |      1 -> 4      |
+------------------+------------------+
```

Figure 4: “us-east1” Region “NVIDIA_P100” GPU quotas confirmation email

Step 2: Setup a Google Cloud VM Instance:

Create an instance with instance name → Choose “us-east1b” as region (Not all regions have GPUs support, the GPU quota has been introduced in step1, Choose the region you applied GPUs) → Choose the number of CPU in the machine type and customize by selecting number and kind of GPUs. → Choose “Ubuntu 16.04 LTS” and increase the disk size in Boot Disk option. → Allow HTTP and HTTPS traffic for your instance.

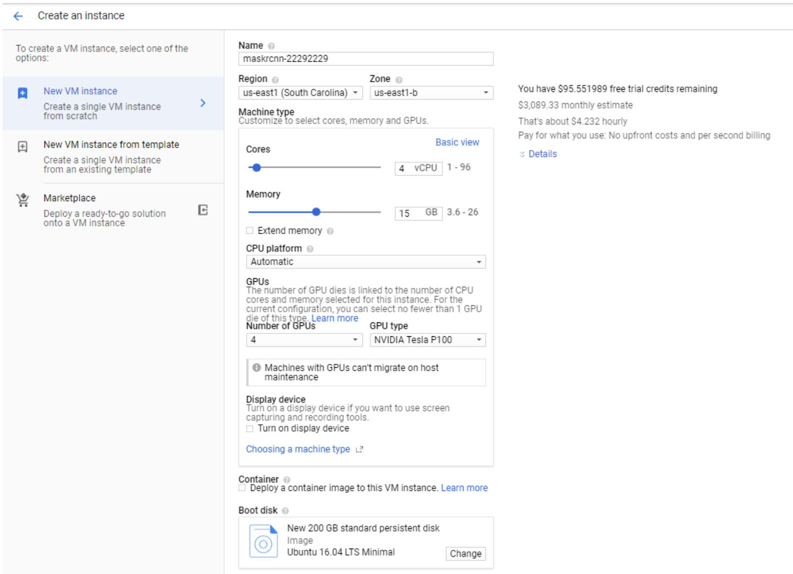


Figure 5: Create VM instance with GPU in Google Cloud Platform

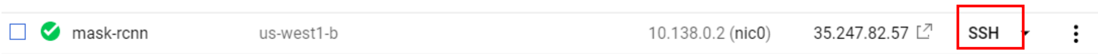
Then a VM instance is created successfully and remember to stop it when you leave the system. The cost can be checked at Bill.

Credits

Promotion ID	Expires	Promotion value	Amount remaining
Free Trial	Apr 14, 2020	$418.25	$95.47

Figure 6: Bill of the project

Step 3: Connect to instance by SSH Server and deployment Weed detection Project: Once the instance is set up and started, click the SSH button to connect with SSH server.



Install Anaconda3 → Install NVIDIA, Cuda9 and cuDNN9-v7.1 library. (an easy bash file to check and install Cuda as following code1) →Install pycocotools with PythonAPI → Install python packages as following code2 and Tensorflow-GPU. (<u>$pip list</u> to check the version of all packages)

```
#!/bin/bash
echo "Checking for CUDA and installing."
# Check for CUDA and try to install.
if ! dpkg-query -W cuda; then
   # The 16.04 installer works with 16.10.
   wget http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64/cuda-
repo-ubuntu1604_9.0.176-1_amd64.deb
   sudo dpkg -i cuda-repo-ubuntu1604_9.0.176-1_amd64.deb
   sudo apt-key adv --fetch-keys
http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64/7fa2af80.pub
   sudo apt-get update
   sudo apt-get install cuda-9.0
fi
echo 'export CUDA_HOME=/usr/local/cuda-9.0' >> ~/.bashrc
echo 'export PATH=$PATH:$CUDA_HOME/bin' >> ~/.bashrc
echo 'export LD_LIBRARY_PATH=$CUDA_HOME/lib64' >> ~/.bashrc
source ~/.bashrc
```

code 1: Bash file to check and install Cuda

```
jiaqihan1108@mask-rcnn-22292229:~$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2017 NVIDIA Corporation
Built on Fri_Sep__1_21:08:03_CDT_2017
Cuda compilation tools, release 9.0, V9.0.176
jiaqihan1108@mask-rcnn-22292229:~$ nvidia-smi
Wed May 15 08:47:33 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.40.04    Driver Version: 418.40.04    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   38C    P0    28W / 250W |    100MiB / 16280MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla P100-PCIE...  Off  | 00000000:00:05.0 Off |                    0 |
| N/A   40C    P0    28W / 250W |      0MiB / 16280MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla P100-PCIE...  Off  | 00000000:00:06.0 Off |                    0 |
| N/A   38C    P0    27W / 250W |      0MiB / 16280MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla P100-PCIE...  Off  | 00000000:00:07.0 Off |                    0 |
| N/A   40C    P0    29W / 250W |      0MiB / 16280MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0      2156      G   /usr/lib/xorg/Xorg                            100MiB |
+-----------------------------------------------------------------------------+
```

Figure 7: Nvidia details

```
pip install numpy
pip install scipy
pip install cython
pip install h5py
pip install Pillow
pip install scikit-image
pip install keras
pip install theano
pip install jupyter
pip install six
pip install opencv-python
pip install imgaug
```

Code 2: the install code for some important packages

Step 4: Connect the system with Jupyter Notebook:

The Jupyter Notebook of the instance is running at http://mask_RCNN:8888/, because the External IP is 35.247.82.57 (it always changes up to the instance) so the link is http:// 35.247.82.57:8888/. "Ctrl + c" to shut down the server. The password of Jupyter Notebook is "hanjiaqi"

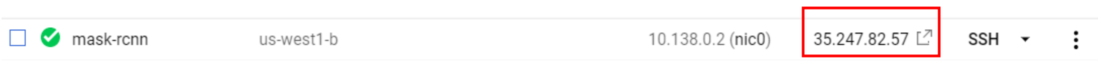


Figure 8: External IP of the instance

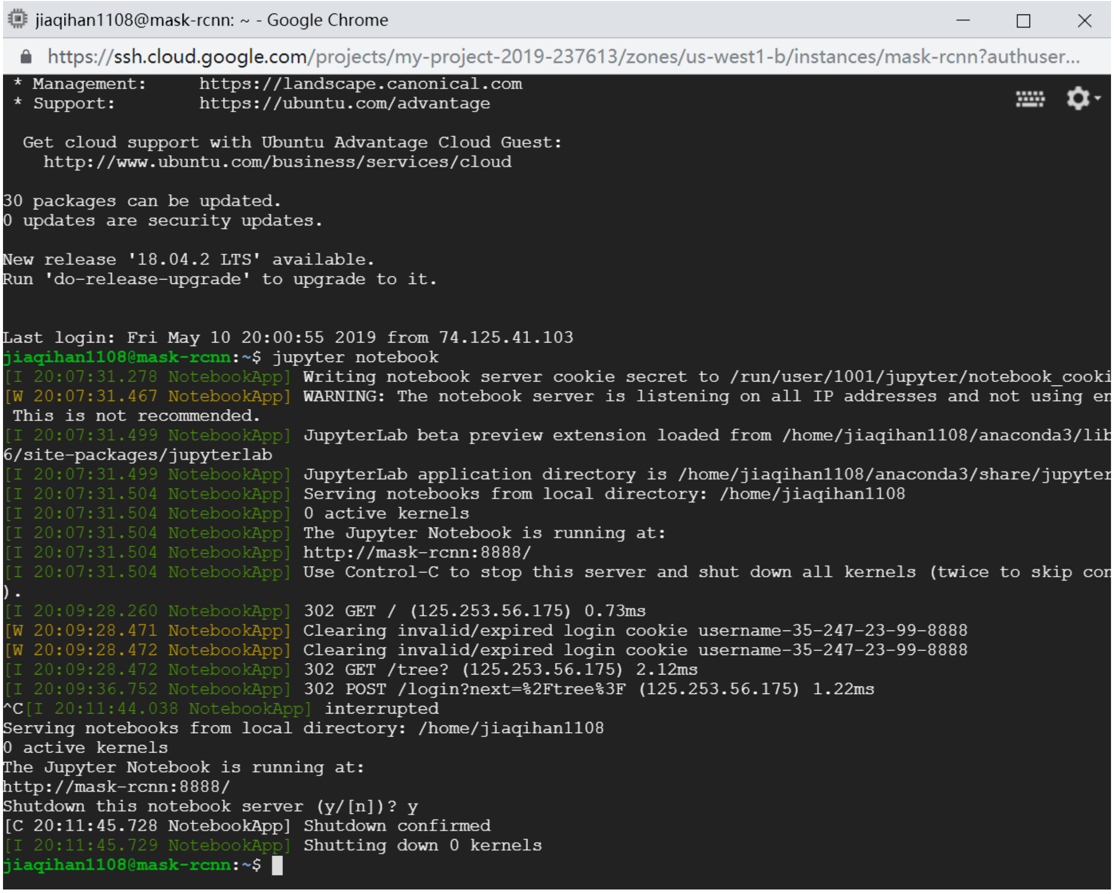


Figure 9: Running Jupyter Notebook of the instance and shut down

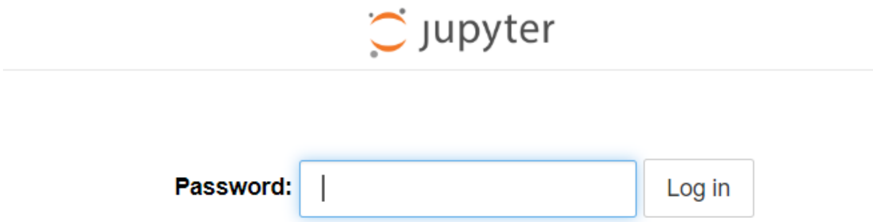


Figure 10: The Password of Jupyter Notebook is “hanjiaqi”

Figure 11: The Jupyter Notebook of the instance and the project interface

Step 5: Create Static external IP for the Instance:

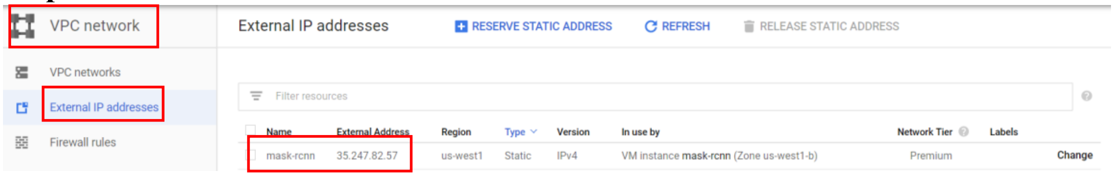


Figure 12: ‘mask-rcnn’ Weed Detection VM instances of Google Cloud platform with a Static External IP

Then this instance has a Static External IP (35.247.82.57) , the final link is always be http:// 35.247.82.57:8888/.

III. Mask_RCNN System Directory Tree.

/Mask_RCNN folder

|__ mrcnn folder: contained the main code for Mask_RCNN algorithm and .h5 model for further retrain.

|__ samples folder: contained Weed_retrain.py and Weed_detection.py

|__ image folder: stored the farmer images

|__ logs folder: saved the retrained model and logs

|__ train_data folder: stored the training data for the model retrain.

|__ …

IV. System Maintenance Problems and Limitations

1. Deployment Problems

1> Compile errors

This kind of errors happened because of the versions of system’s Python packages do not match with the versions of anaconda Python. And when compile the Makefile, it will use the system’s Python as default.

The solution is to replace the system’s Python with Anaconda Python. For example, use Cython-0.28.1 (/home/jiaqihan1108/anaconda3/bin/python setup.py build_ext install)

Figure 13: Change the path of python for compiling

2> Make sure all packages’ version matched.

2. Quota Allocation (Resource Limitation)

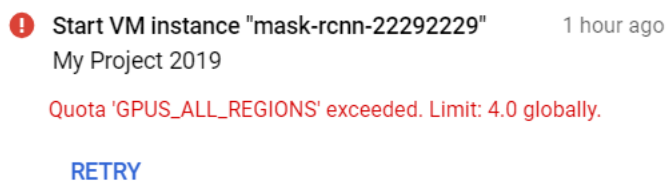


Figure 14: GPU limit of all regions

This error happened because of the unreasonable resource allocation. There is a limitation of GPUs of all regions. Sometimes we have two or more instances running together, the total number of these instances' GPU cannot over 4 globally.

The recommend solution is to deploy and balance the workload across multiple zones or regions to reduce the likelihood of an outage. So never allocate all GPU resources in one instance since sometimes we need to run more than one instance.

3. VM Instances Dead and Storage (Platform Limitation)

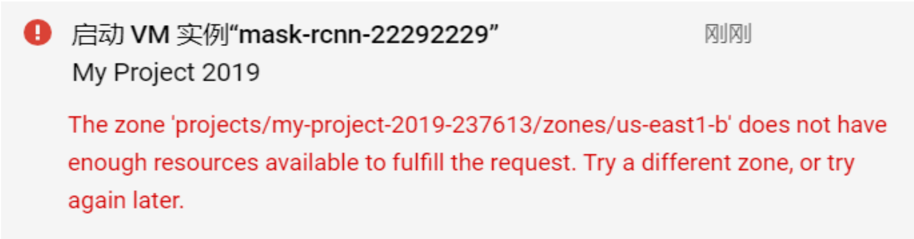


Figure 15: Instance dead

The reason of this kind of error is because the Google Cloud Platform is not as stable as we want.

The solution is to use snapshot to recreate instances. If something wrong happened with the instance, we can recreate the instance through its snapshot. A snapshot should be created once the instance is upgraded to a new version or implemented new functions.

Step details:

Step 1: Create a snapshot of the instance mask-rcnn

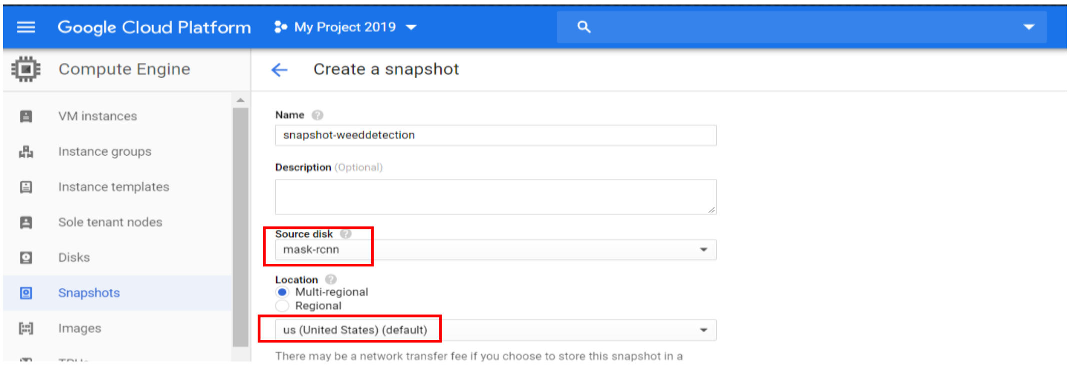


Figure 16: Create a Snapshot of mask-rcnn instance

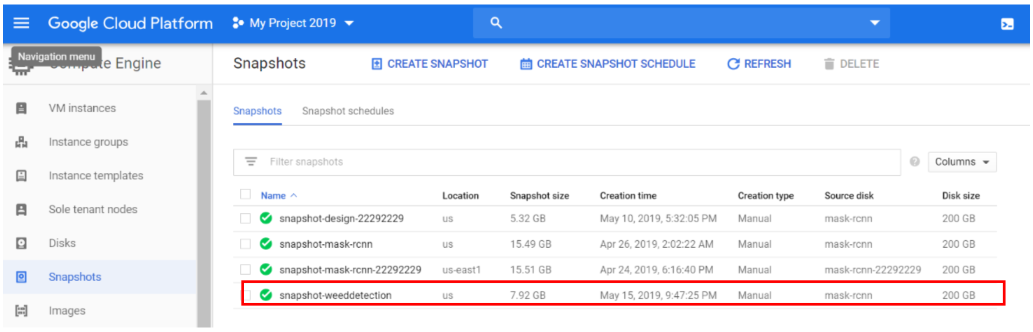


Figure 17: Snapshots of all milestones during the development

Step 2: Create a new instance based on the snapshot

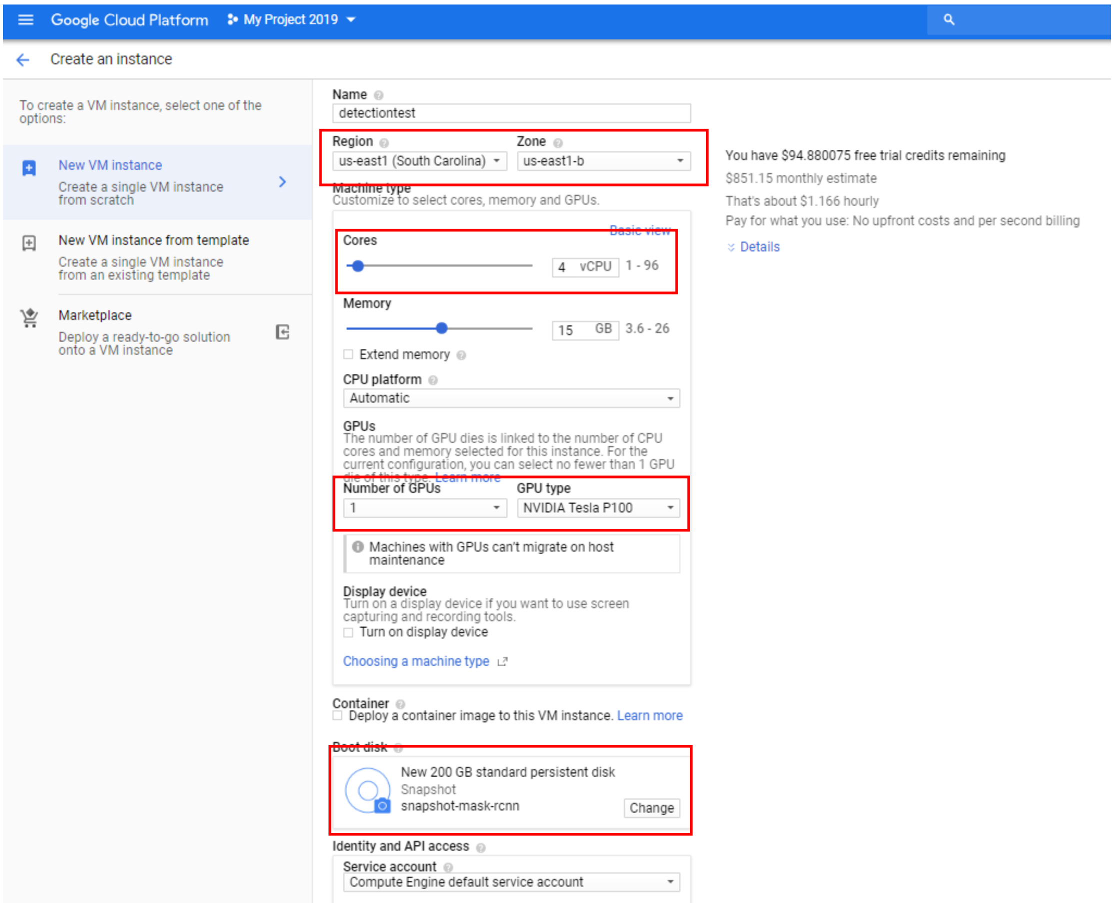


Figure 18: Create a instance based on the snapshot

Remember to allow HTTP and HTTPS traffic for your instance

Name	Zone	Recommendation	In use by	Internal IP	External IP	Connect
detectiontest	us-east1-b			10.142.0.13 (nic0)	34.73.82.207	SSH
fcn-22292229	us-east1-b			10.142.0.8 (nic0)	None	SSH
mask-rcnn	us-west1-b			10.138.0.2 (nic0)	35.247.82.57	SSH
mask-rcnn-22292229	us-east1-b			10.142.0.9 (nic0)	34.74.74.248	SSH

Figure 19: A new instance "detectiontest" same with instance "mask-rcnn"

4. Model Retrain (Platform Limitation)

1> The python file cannot run.
That because the mode of the python file should be changed.
Use "chmod a+x Weed_retrain.py" and directly run the python file "./Weed_retrain.py".

2> OS Error: can't allocate memory
That because the system cannot support Multi-threading perfectly.
The solution is to make sure the system running with single multithread and let "worker = 1" in model.py. (in the mrcnn folder) As below:

```
            self.keras_model.fit_generator(
                train_generator,
                initial_epoch=self.epoch,
                epochs=epochs,
                steps_per_epoch=self.config.STEPS_PER_EPOCH,
                callbacks=callbacks,
                validation_data=val_generator,
                validation_steps=self.config.VALIDATION_STEPS,
                max_queue_size=100,
                #workers=workers,
                workers=1,
                use_multiprocessing=False,
                # use_multiprocessing=True,
           )
            self.epoch = max(self.epoch, epochs)
```

Code 3: Use single threading

5. Detection result (Development Limitation)

The system only can automatically show the detection image on the Jupyter Notebook page but cannot be saved in local folder automatically. The only way to save the results is right-click and save the result image one by one.

V. Further Development – FCN System

1. Deployment Requirement

Python3.5（Anaconda3） + tensorflow-gpu1.4.0 + Cuda 8.0 +cudnn-8.0-2.windows10-x64-v6.0

2. Training dataset

224*224 images and their 8bit mask in 2 separate folders (Image and Mask).

3. FCN System Directory Tree

/FCN folder

|__ Model folder: the Vgg19 model for further retrain
|__ samples folder: contained Weed_retrain.py and Weed_detection.py
|__ images folder: stored the farmer images
|__ logs folder: saved the retrained model and logs
|__ Data_zoo folder: stored the training data for the model retrain.
|__ FCN.py: main function to retrain the model and achieve detection
|__ …

4. System Maintenance Problems and Limitations

1. training loss is not good (Training Dataset Limitation)

The FCN algorithm needs a large training dataset for training own model, but we don't have too much weed images with its masks.