Google Cloud Setup for Pytorch with GPU

September 10, 2019

Author: Thamme Gowda *tg@isi.edu*

1 Create a Virtual Machine using a Prebuilt Image for Pytorch

Goto this page

https://console.cloud.google.com/marketplace/config/click-to-deploy-images/tensorflow
Configure the virtual machine settings as follows:

1. Deployment Name: nlp-class
2. vCPUs 13GB Memory (Default)
3. Num GPUs 1 ; GPU Type Tesla K80
4. Framework Pytorch 1.2 + fast.ai Cuda 10.0
5. Check "GPU install NVIDIA GPU driver auto on first statup"
6. Check"Access to the Jupyter Lab Enable access to JupyterLab via URL instead of SSH. (Beta)"
7. Create

When it is done, Click onn "SSH" option to get a ssh shell

1. `nvidia-smi` shows GPU
2. `python -c 'import torch; print(torch.cuda.is_available())'` prints `True` when everything setup correctly
3. `curl ifconfig.me` prints external IP address of VM. Notedown this for step 4.
4. `jupyter lab --ip=0.0.0.0 --port=8888` starts jupyter lab on port 8888. `--ip=0.0.0.0` is to make it accessible from other machines (such as your laptop). It also prints a token on for authentication, notedown for step 4.
5. On your web browser goto http://<IP-from-setp3>:8888 . It should give you a box for pasting token. Paste the token noted from step 4. You should have access to Jupyter Notebook with these.

---

2 Creating a VM manually and configuring it.

I didnt know there was a prebuilt image. I created this tutorial before finding the prebuilt image. Sharing this incase someone needs it for customising the runtime

2.1 Overview

- Google Compute Cloud -- for the hardware (CPU/RAM/GPU/Stoarge)
- Conda -- Installing requitements and managing environments
- Cuda and Nvidia libs for GPU acceleration
- Jupyter lab/notebook -- Interactive python shell with documentation/notes
- and ofcourse, **PyTorch** -- the main focus

3 Google Compute Cloud

One of the places where you can rent some compute+storage hardware.

_**Disclaimer**: Am I selling Google services to you? No! We will need GPUs for deep learning, and they have them, and they were kind enough to give free credits. __

3.1 Entities:

- Project
  - Instance (CPU + RAM + GPU)
    - Operating System -- Recommended: some variant of linux with GNU tools, eg: Ubuntu . or Centos
    - Persistent Disk: where you store data
    - Network Interfaces: protect or expose node
- Billing

Note: In this era of cloud services, emphasize on *data* and *code* of your research/homework. Why? You can almost always trade other components(CPU/GPU etc) with money. But *data+code* (both of them reside on *a persistant disk*) is more valuable than what money+time can buy.

3.1.1 Google Compute Console

3.2 Create Project

- Go to https://console.cloud.google.com/cloud-resource-manager
- Click "Create Project" on the top left
- Give a name -- for example "nlpclass" -- click "create"

Now onwards, remember to have your project selected from the drop down next to "Google Cloud Platform" on the top left (on the bar).

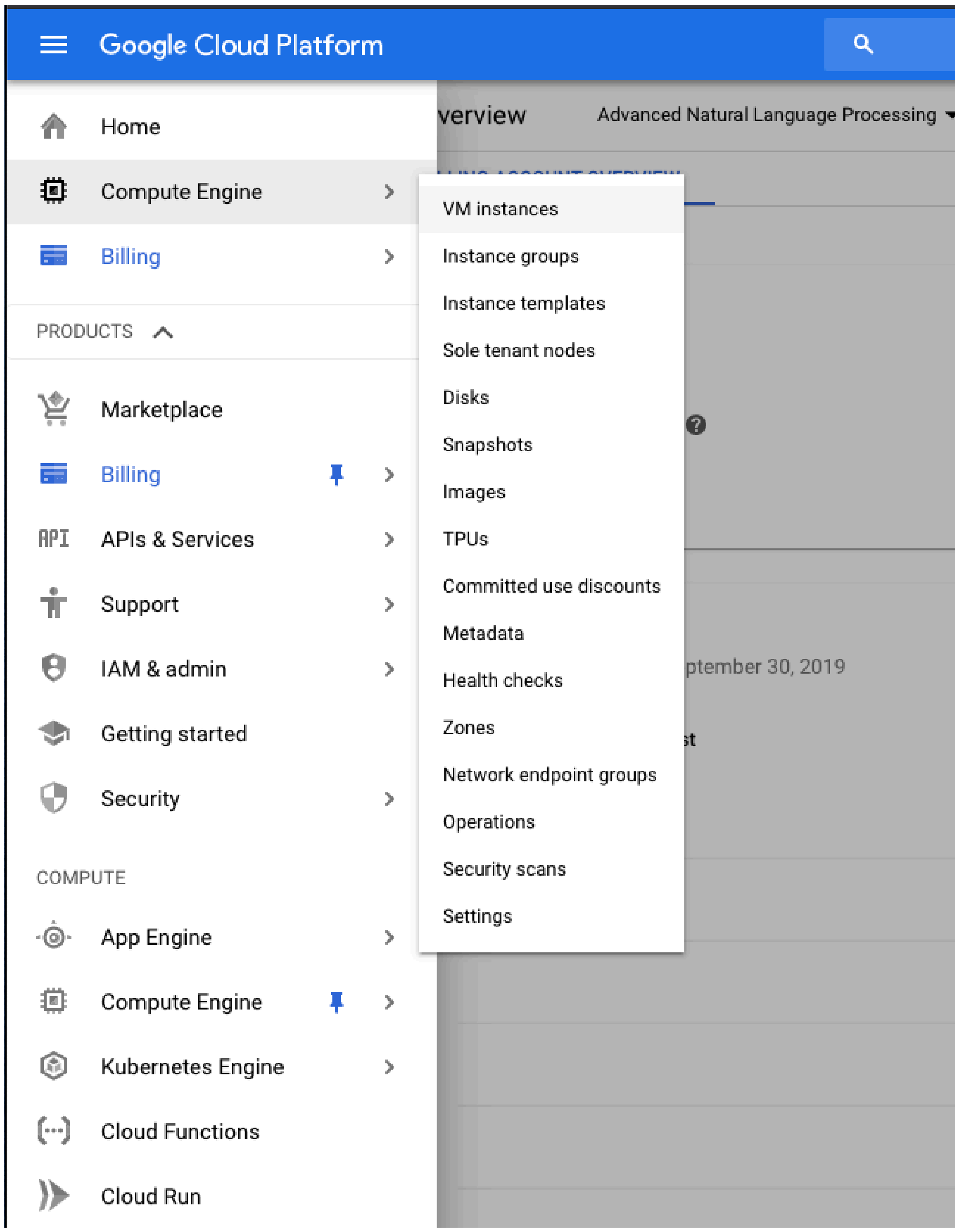
Note: If GCC ever asks to enable Google Compute Engine API, please enable it.

3.3 Estimate before you rent

Cost Estimator https://cloud.google.com/products/calculator/

Note: 1. the GPU+CPU costs only when you keep the machine running. But storage costs irrespecitve of whether you use it or not.
2. We are NOT trying to get top end GPU or CPU or storage for since they are often very expensive and we dont need them for class homework

Here is an estimate:


Google Cloud Platform
Home
Compute Engine
Billing
PRODUCTS
Marketplace
Billing
APIs & Services
Support
IAM & admin
Getting started
Security
COMPUTE
App Engine
Compute Engine
Kubernetes Engine
Cloud Functions
Cloud Run
VM instances
Instance groups
Instance templates
Sole tenant nodes
Disks
Snapshots
Images
TPUs
Committed use discounts
Metadata
Health checks
Zones
Network endpoint groups
Operations
Security scans
Settings
verview
Advanced Natural Language Processing
ptember 30, 2019
st

1. GPU; Nvidia Tesla K80s (the low end, cheapest GPU)

- GPUs are costrained: not available on all Regions/Zones and not compatible with all CPU setups. If you are going to use GPU, try to get this constraint sorted out first.

2. CPU either 2 core or 4core CPU, with 16GB RAM
3. 200GB standard persistant stoarge (Not the SSD; dont think we need SSD)

With GPU

Suggestions for reducing cost: 1. GPU is the most expensive resource as of now 2. Development takes time, You dont need GPU for development; You can do it on CPU + Try to do this on your laptop if possible! or Google Colab 3. Once you have working code to run, rent a GPU and run it on GPU. GPU is faster 4. Keep the storage peristant -- remember that the code+data you have is more valuable asset than the rest. + Take backup of code. Suggestion: not the googledrive or dropbox, but git/github for the code

3.4 Create a Virtual Machine instance

Got to https://console.cloud.google.com/compute/instances

Find an option to "Create".

Here is an example

3.4.1 OS and disk.

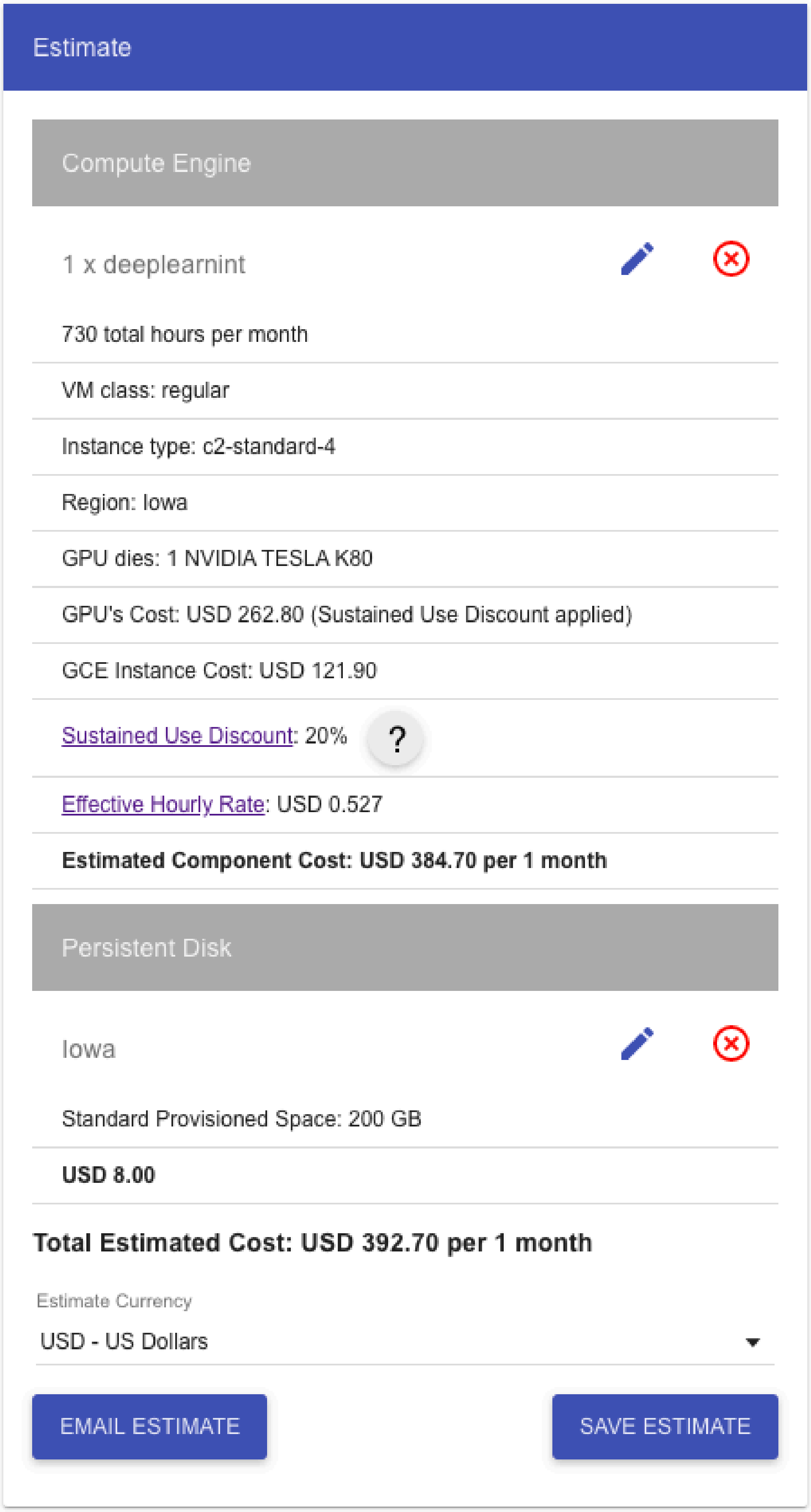
1. For the sake of simplicity, we are going to use one disk for Boot and Data, hence increased boot disk size from 10GB to 200GB. In practice you would want to seperate Boot disk and Data disk. However, for a beginner, that setup creates more complications and confusions (If this doesnt make sense, dont worry, its fine. If you know what you are doing go and make two disks).
2. In this setup, remember to Uncheck "Delete Boot Disk when instance is deleted".

3. Why CentOS 7 instead of ___ ? USC HPC runs it, so better be familiar wit it.

Thats it, create! This is going to take some time.

> You might get error message: Quota 'GPUS_ALL_REGIONS' exceeded. Limit: 0.0 globally.
> Here is how to fix it https://stackoverflow.com/a/53678838/1506477 . By default GPU quota is set to 0, we need to set it to 1. (I think 0 means disabled, 1 means 100%)
> Find the Retry button to resend the VM creation request

If GPUs are constrained resource, they may not be available when you try, you might get an error. What can we do then? Two options: 1. Go back and create the similar machine without GPU; Maybe consider adding more CPU cores like 4 instead of 2 2. IF GPU is needed, try again at some other time, or switch datacenter zone



Estimate

Compute Engine

1 x deeplearnint

730 total hours per month

VM class: regular

Instance type: c2-standard-4

Region: Iowa

GPU dies: 1 NVIDIA TESLA K80

GPU's Cost: USD 262.80 (Sustained Use Discount applied)

GCE Instance Cost: USD 121.90

Sustained Use Discount: 20% ?

Effective Hourly Rate: USD 0.527

Estimated Component Cost: USD 384.70 per 1 month

Persistent Disk

Iowa

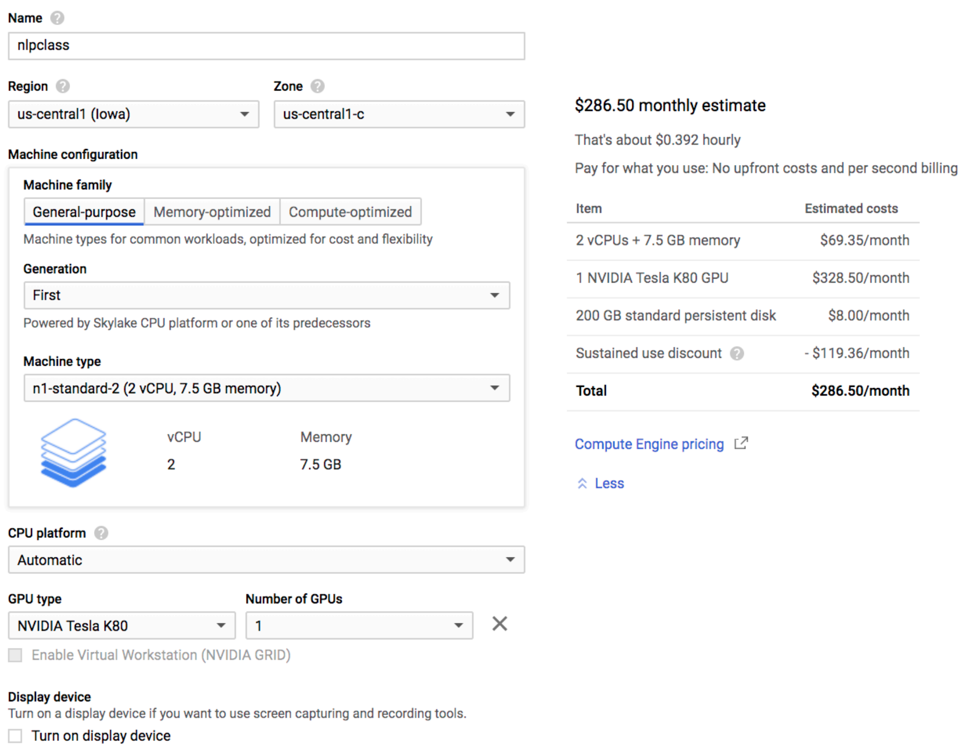
Standard Provisioned Space: 200 GB

USD 8.00

Total Estimated Cost: USD 392.70 per 1 month

Estimate Currency

USD - US Dollars



Name
nlpclass

Region: us-central1 (Iowa)
Zone: us-central1-c

Machine configuration

Machine family
[✓ General-purpose] Memory-optimized Compute-optimized
Machine types for common workloads, optimized for cost and flexibility

Generation
First
Powered by Skylake CPU platform or one of its predecessors

Machine type
n1-standard-2 (2 vCPU, 7.5 GB memory)

vCPU	Memory
2	7.5 GB

CPU platform
Automatic

GPU type: NVIDIA Tesla K80
Number of GPUs: 1

☐ Enable Virtual Workstation (NVIDIA GRID)

Display device
Turn on a display device if you want to use screen capturing and recording tools.
☐ Turn on display device

$286.50 monthly estimate

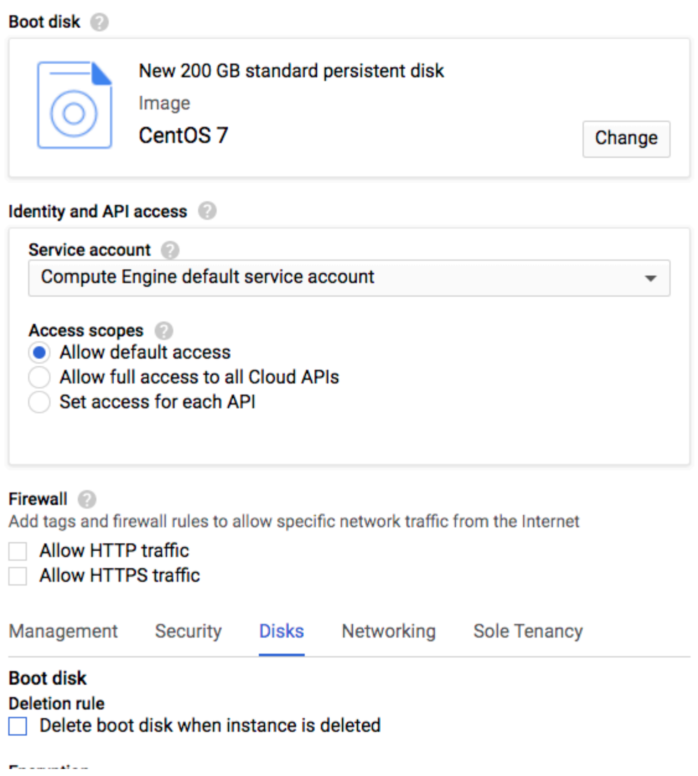
That's about $0.392 hourly

Pay for what you use: No upfront costs and per second billing

Item	Estimated costs
2 vCPUs + 7.5 GB memory	$69.35/month
1 NVIDIA Tesla K80 GPU	$328.50/month
200 GB standard persistent disk	$8.00/month
Sustained use discount	- $119.36/month
Total	**$286.50/month**

Compute Engine pricing

Less



Boot disk

New 200 GB standard persistent disk
Image
CentOS 7
Change

Identity and API access

Service account
Compute Engine default service account

Access scopes
[✓ Allow default access]
Allow full access to all Cloud APIs
Set access for each API

Firewall
Add tags and firewall rules to allow specific network traffic from the Internet
☐ Allow HTTP traffic
☐ Allow HTTPS traffic

Management Security **Disks** Networking Sole Tenancy

Boot disk
Deletion rule
☐ Delete boot disk when instance is deleted

3.5 Login to the virtual machine

Now that you got some machine rented on Google Cloud, time to go inside and do stuff. Here https://console.cloud.google.com/compute/instances you will see cerated instance. A Green Tick indicates it is running.

Clicking on the SSH should give you instance ssh in browser. If you are happy with it, you can work directly from browser. But most of us prefer our own terminal emulator clients such as iterm (OSX), Terminator (linux) or PuTTY (windows). We can do that too, I leave that to your own exploration.

`df -h` should show disk space

3.6 Practice Starting Stoping Instance

https://console.cloud.google.com/compute/instances

3.7 External IP

https://console.cloud.google.com/compute/instances
When instance is running, you will see External IP. This is where you can get External IP. You can also assign static IP (which is beyond the scope of this tutorial) Alternatively, when you are logged inside the VM, `curl ifconfig.me` to see its external IP.

3.8 Updating Resources of Virtual Machine (After Creation)

Stop the machine : Goto https://console.cloud.google.com/compute/instances > Select > Stop (or shutdown)

Once stopped, click on the instance name > (a new page shows) > Edit (here you are free to edit the resources).

If you want to remove GPU, this is the place. Just click close mark "X" next to GPU and save

3.8.1 Allow Network Ports

We are going to use Jupyter lab which is a web service that uses port 8888. By default google cloud blocks all ports. Let us open port . This page has relevant details: https://docs.bitnami.com/google/faq/administration/use-firewall/

Allow tcp port 8888 from anywhere.

VM instance details > Network interfaces "View Details" > "Firewall Rules" (on the left side, 3rd option) > "Create Firewall Rule" 1. Enter Name "allow8888", similary description 2. Direction "Ingress" Action to match "Allow" 3. Target: "All instances in network" 4. Source Filter "IP Ranges" Source IP ranges "0.0.0.0/0" 5. Specified protocols and ports: Check "tcp" adn enter "8888" 6. "Create"

---

4 Software Installation

`mkdir work && cd work` - create a work directory, this is the directory you need to backup when times comes to close the lease

Install wget and tmux : `sudo yum install wget tmux htop`

Recommended (not mandatory) `tmux` is a very useful tool when working with remote machines. We wont teach it, but highly recommended for increasing the productivuity (i.e do more quicker).

```
tmux              # creates a new tmux ession
tmux ls           # lists all available sessions
tmux a -t <target_number>  # attach to the desired target tmux
# to detach: when in tmux session type 'CTRL+b d'
# tmux by default uses prefix CTRL+b
```

Install your favorite editor, I am going to do `sudo yum install emacs-nox`

4.1 Install Conda

Conda is a popular environment manager, package installer that works across platforms: linux/Mac/Windows. Conda comes in two flavors: miniconda and anaconda. We want miniconda and NOT anaconda. Get the URL of latest miniconda for linux 64-bit && python 3.7 from https://docs.conda.io/en/latest/miniconda.html

```
cd ~/work
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh
```

Read the instructions from installer; review the license, and 'yes' to approve. leave the default settings (unless you know what you are doing)

When the installer finishes, it would ask: >installation finished. Do you wish the installer to initialize Miniconda3 by running conda init? [yes|no]

You should say 'yes'

conda edited environment settings, particularly bashrc. You need to reload bashrc. Ideally quit the current bash session and get a new bash session with new environment. For now just type `bash` command, it will do the same.

Test conda

```
conda info
conda env list
$ python
Python 3.7.3 (default, Mar 27 2019, 22:11:17)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
```

We are almost there!

Pro mode: create a separate conda environment for project. For the beginner, conda already created a default environment called `base`, and we are going to use the same for this project.

4.2 Setup GPU drivers

Caution: Getting nvidia drivers work well is a rabbit hole.

Google has given a script: https://cloud.google.com/compute/docs/gpus/add-gpus#install-driver-script

Copy paste this script (which I copied from that link) to a script file, say `install-cuda.sh`

```
#!/bin/bash
# Install kernel headers and development packages
echo "Installing kernel headers and development packages."
yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
echo "Checking for CUDA and installing."
# Check for CUDA and try to install.
if ! rpm -q cuda-10-0; then
  curl -O http://developer.download.nvidia.com/compute/cuda/repos/rhel6/x86_64/cuda-repo-rhel6-
  rpm -i --force ./cuda-repo-rhel6-10.0.130-1.x86_64.rpm
  yum clean all
  # Install Extra Packages for Enterprise Linux (EPEL) for dependencies
  yum install epel-release -y
  yum update -y
  yum install cuda-10-0 -y
fi
# Verify that CUDA installed; retry if not.
if ! rpm -q cuda-10-0; then
  yum install cuda-10-0 -y
fi
```

Then run `sudo bash install-cuda.sh` This is going to take lot of time. Once finished, run

```
$ nvidia-smi
Tue Sep 10 22:47:44 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.87.00    Driver Version: 418.87.00    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 K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   59C    P0    82W / 149W |      0MiB / 11441MiB |    100%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Okay, now we have a GPU running! Almost there, we just need to install Pytorch.

4.3 Install Pytorch

Copy command from https://pytorch.org/get-started/locally/

Note, we have cuda 10.1, specify that version correctly

```
conda install pytorch torchvision cudatoolkit=10.1 -c pytorch
```

Once the installer finishes, check if torch agrees that GPU is available for use

```
python -c 'import torch; print(torch.cuda.is_available())'
```

4.4 Jupyter Lab (previously Jupyter Notebook)

https://github.com/jupyterlab/jupyterlab

```
$ conda install -c conda-forge jupyterlab

$ mkdir ~/work/pytorch-nlp
$ cd ~/work/pytorch-nlp
$ jupyter lab
```

5 Getting Started with Pytorch for NLP

```
$ mkdir ~/work/pytorch-nlp
$ cd ~/work/pytorch-nlp
$ jupyter lab
```

To be Continued with Another notebook