Uninstall existing anaconda

You might prefer miniconda (a small size version of anaconda) over anaconda
In this case, you need to uninstall anaconda first.
Please refer to : https://docs.anaconda.com/anaconda/install/uninstall/

$rm -rf ~/anaconda3
$rm -rf ~/.condarc ~/.conda ~/.continuum

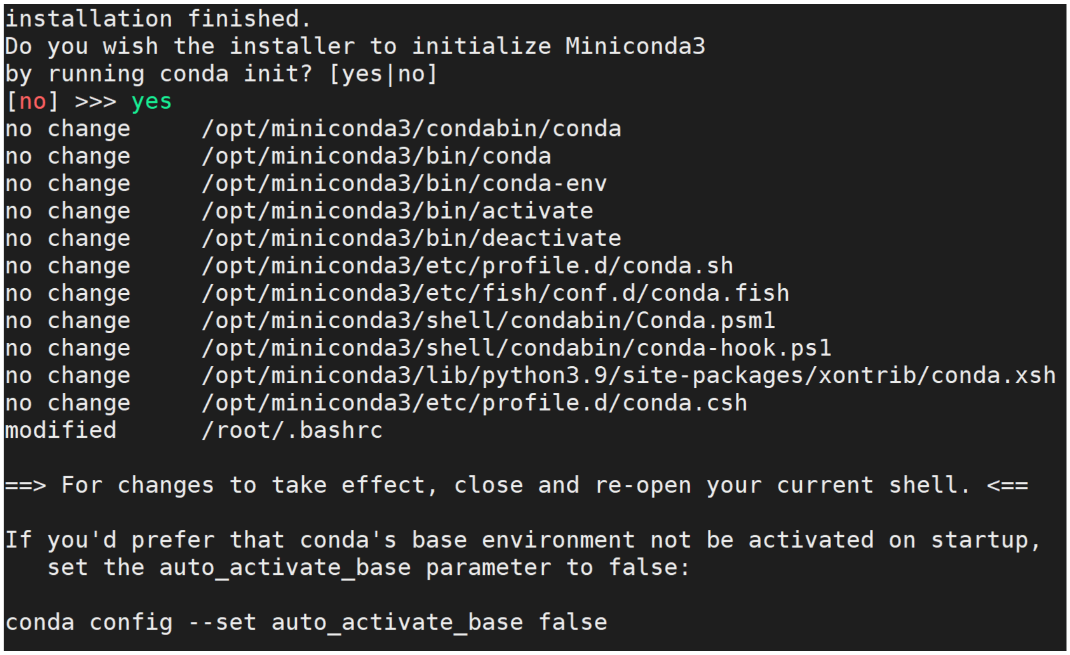
Install miniconda (on RHEL 8)

Download miniconda
https://docs.conda.io/en/latest/miniconda.html

Install for multi-user
Refer to : https://docs.anaconda.com/anaconda/install/multi-user
$chmod +x Miniconda3-latest-Linux-x86_64.sh

Install with sudo
$sudo bash Miniconda3-latest-Linux-x86_64.sh

During installation, choose "/opt/miniconda3"
[/root/miniconda3] >>> /opt/miniconda3



```
installation finished.
Do you wish the installer to initialize Miniconda3
by running conda init? [yes|no]
[no] >>> yes
no change     /opt/miniconda3/condabin/conda
no change     /opt/miniconda3/bin/conda
no change     /opt/miniconda3/bin/conda-env
no change     /opt/miniconda3/bin/activate
no change     /opt/miniconda3/bin/deactivate
no change     /opt/miniconda3/etc/profile.d/conda.sh
no change     /opt/miniconda3/etc/fish/conf.d/conda.fish
no change     /opt/miniconda3/shell/condabin/Conda.psm1
no change     /opt/miniconda3/shell/condabin/conda-hook.ps1
no change     /opt/miniconda3/lib/python3.9/site-packages/xontrib/conda.xsh
no change     /opt/miniconda3/etc/profile.d/conda.csh
modified      /root/.bashrc

==> For changes to take effect, close and re-open your current shell. <==

If you'd prefer that conda's base environment not be activated on startup,
   set the auto_activate_base parameter to false:

conda config --set auto_activate_base false
```

check available groups
$cat /etc/group

add conda group where you can add user to use miniconda
$sudo groupadd conda
$cat /etc/group | grep conda

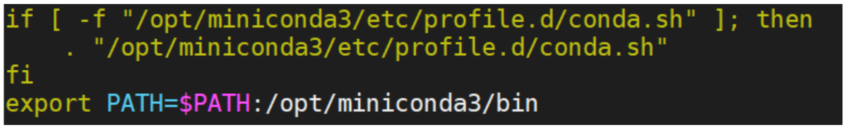
Change the group ownership to “conda” on the entire directory where Anaconda is installed [1]
$sudo chgrp -R conda   /opt/miniconda3

Set read and write permission for the owner, root, and the “conda” only:
$sudo chmod 770 -R /opt/miniconda3/

Add ***TargetUser*** to a group. Users added to the “conda” group now have the ability to access Anaconda, install packages, and create environments.
$sudo usermod -a -G conda <***TargetUser***>

Log out and log back.

Edit ~/.bashrc, add the following lines for miniconda



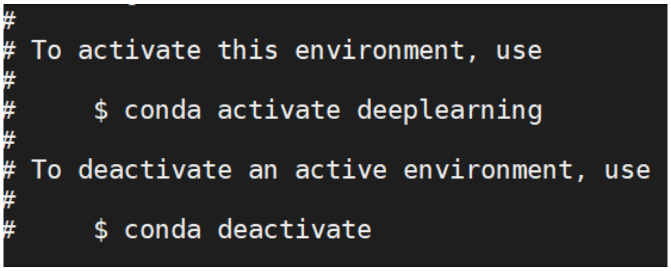
```
if [ -f "/opt/miniconda3/etc/profile.d/conda.sh" ]; then
    . "/opt/miniconda3/etc/profile.d/conda.sh"
fi
export PATH=$PATH:/opt/miniconda3/bin
```

Update the bashrc
$source ~/.bashrc

You can check install the conda version.
$ conda --version
conda 4.10.3

Install tensorlfow-gpu

Create an env for deep learning work
Use python 3.9
$conda create -n deeplearning python=3.9



```
#
# To activate this environment, use
#
#     $ conda activate deeplearning
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

To check the env list,
$conda env list

To remove
$conda remove --name deeplearning --all

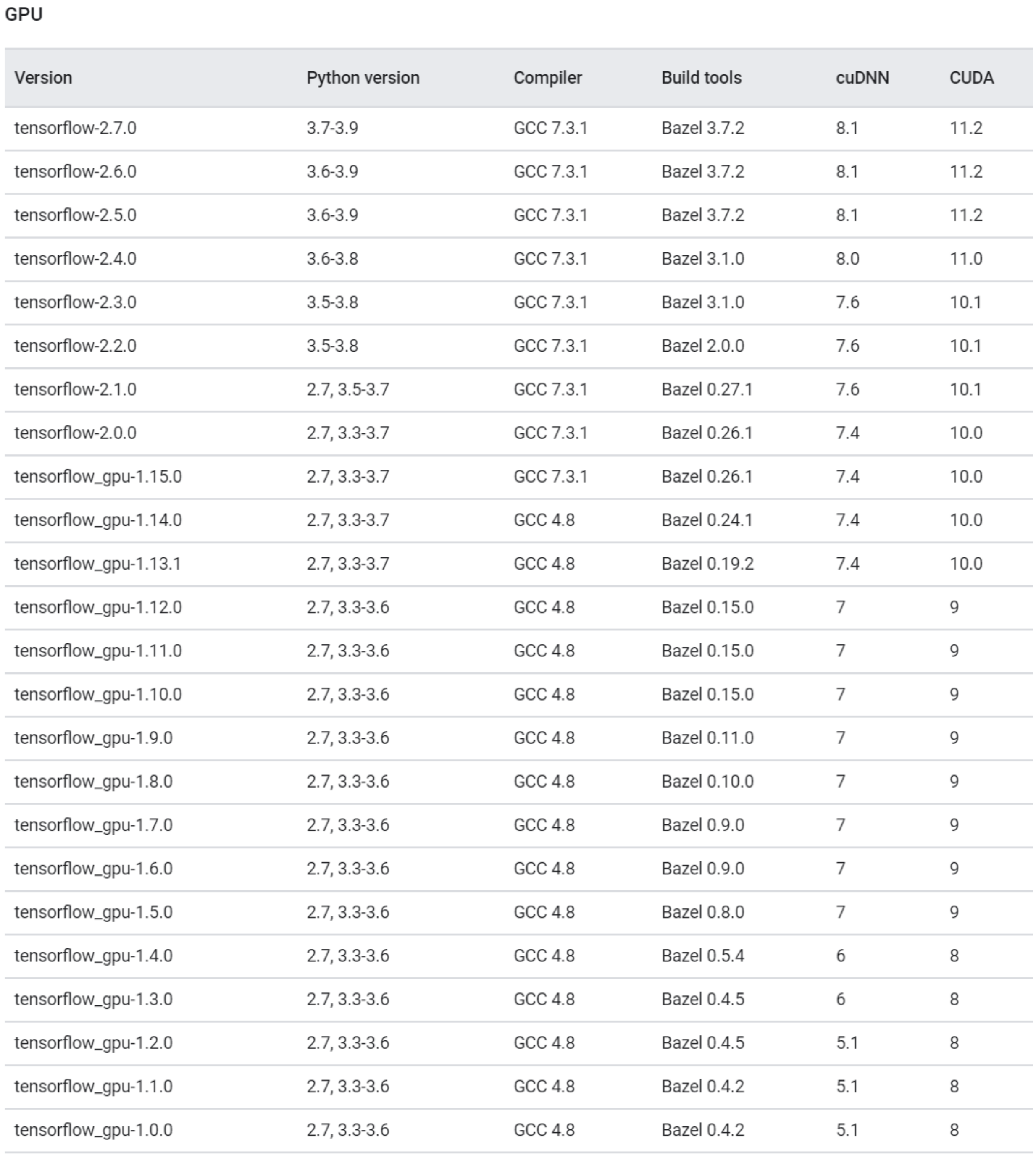
To list the packages for the deepleaning env,
$conda list -n deeplearning

Install tensorflow-gpu
In the deeplearning env, run
$conda install pip

Install tensorfow-gpu 2.7.0 on rehl 8
$pip install
https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow_gpu-2.7.0-cp39-cp39-manylinux2010_x86_64.whl

For different whl file, check the following link
https://www.tensorflow.org/install/pip#conda

To make sure tensorflow with gpu support will run on your NVIDIA GPUs, you will need to install the GPU driver and CUDA toolkit. You will need to set up an NVIDIA developer account and download the corresponding cudnn libraries as well. Noted that, make sure your installed versions are supported for the target tensorflow version.



GPU

Version	Python version	Compiler	Build tools	cuDNN	CUDA
tensorflow-2.7.0	3.7-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.6.0	3.6-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.5.0	3.6-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.4.0	3.6-3.8	GCC 7.3.1	Bazel 3.1.0	8.0	11.0
tensorflow-2.3.0	3.5-3.8	GCC 7.3.1	Bazel 3.1.0	7.6	10.1
tensorflow-2.2.0	3.5-3.8	GCC 7.3.1	Bazel 2.0.0	7.6	10.1
tensorflow-2.1.0	2.7, 3.5-3.7	GCC 7.3.1	Bazel 0.27.1	7.6	10.1
tensorflow-2.0.0	2.7, 3.3-3.7	GCC 7.3.1	Bazel 0.26.1	7.4	10.0
tensorflow_gpu-1.15.0	2.7, 3.3-3.7	GCC 7.3.1	Bazel 0.26.1	7.4	10.0
tensorflow_gpu-1.14.0	2.7, 3.3-3.7	GCC 4.8	Bazel 0.24.1	7.4	10.0
tensorflow_gpu-1.13.1	2.7, 3.3-3.7	GCC 4.8	Bazel 0.19.2	7.4	10.0
tensorflow_gpu-1.12.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.15.0	7	9
tensorflow_gpu-1.11.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.15.0	7	9
tensorflow_gpu-1.10.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.15.0	7	9
tensorflow_gpu-1.9.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.11.0	7	9
tensorflow_gpu-1.8.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.10.0	7	9
tensorflow_gpu-1.7.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.9.0	7	9
tensorflow_gpu-1.6.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.9.0	7	9
tensorflow_gpu-1.5.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.8.0	7	9
tensorflow_gpu-1.4.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.5.4	6	8
tensorflow_gpu-1.3.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.5	6	8
tensorflow_gpu-1.2.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.5	5.1	8
tensorflow_gpu-1.1.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.2	5.1	8
tensorflow_gpu-1.0.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.2	5.1	8

Refer to: https://www.tensorflow.org/install/source

After installing the GPU driver and CUDA toolkit, in your ~/.bashrc file, export cuda libraries as shown below.

```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64
```

To install CUDNN, unzip the download files, copy files to the /usr/local/cuda-10.1/include and lib64

```
$sudo scp -rp cuda/include/cudnn.h  /usr/local/cuda/include/
$sudo scp -rp cuda/lib64/libcudnn*  /usr/local/cuda/lib64/
```

Besides that, you can need to install your needed packages and software.

```
$conda install -c conda-forge pydicom
$conda install matplotlib scikit-image scikit-learn
```