Homework 2

Background

In this homework, you will learn a lot on graph representation learning through reading and practising by CogDL toolkit (https://github.com/thudm/cogdl). CogDL is a graph representation learning toolkit that allows researchers and developers to easily train and compare baseline or custom models for node classification, link prediction and other tasks on graphs.

Experiments

1. (10 points) Read the DeepWalk paper (https://arxiv.org/abs/1403.6652), which is the representative of graph embedding methods, and **write a summary** in the report. You can write the motivation, the method, the experiments, or your own ideas about it.
2. (10 points) Read the GCN paper (https://arxiv.org/abs/1609.02907) and **write a summary** in the report. You can write the motivation, the method, the experiments, or your own ideas about it. You should also write the comparison between DeepWalk and GCN.
3. (20 points) Fork & Clone the CogDL (https://github.com/thudm/cogdl) to your machine. You can read the installation instructions in the README to help you install the CogDL. After the installation, **give the results** of running the two following commands (may take a few minutes). You may set --*cpu* when training GCN if you have no CUDA in your machine.

```
python scripts/train.py --task unsupervised_node_classification --
dataset wikipedia --model deepwalk

python scripts/train.py --task node_classification --dataset citeseer --
model gcn
```

4. (20 points) Read the code of CogDL toolkit and see how the code is organized for training different models on different datasets. **Write a summary** about the organization and API design of the code. You can give a figure if necessary.
5. (30 points) **Choose one: (if we choose b or c, you are expected to get better results than the current implementation in the CogDL)**
    a. Implement a GNN model in the CogDL. You should first fork the CogDL toolkit to your own github repository and then **submit pull requests** to the *thudm/cogdl* when you finish implementation. You can choose one of GNN papers in the **appendix A**, or other recent GNN paper that you think is impressive. You should **give the results** of the GNN model using the CogDL toolkit and **compare them** with the original results in the paper. You also need to add unit tests for your implemented GNN model.
    b. Re-implement an existing GNN model in the CogDL without the dependency of PyTorch Geometric. You should first fork the CogDL toolkit to your own github repository and then **submit pull requests** to the *thudm/cogdl* when you finish implementation. You can choose one of GNN papers in the **Appendix B**. You should **give the results** of the GNN model and **compare them** with the results of original implementation in the CogDL. You also need to add unit tests for your implemented GNN model.
    c. Improve existing models on several datasets of one task and compare the results with our leaderboards in the README. For example, for node classification task, you can try to improve the results of GCN and GAT models on three datasets. You can improve the implementation and find more suitable hyper-

parameters for models to improve the results. **Give the results** of all attempts (hyper-parameter configurations) you tried and **compare them**.

6. (10 points) **Write your suggestions** on the CogDL toolkit.
7. (Bonus points) You can contribute to the CogDL in other ways through pull requests before the homework deadline, including:
    a. Write some tutorials in the documents
    b. Add some unit tests to improve the coverage
    c. Add model serving module (may refer to *bert-as-service*)
    d. Design and add visualization demo (may refer to TF playground)
    e. Other content you suggest

Submission

You should submit **one report**, including:

1) Your summaries
2) Experimental results and analyses
3) The #ID and link of your pull request (if exists)
4) Your suggestions
5) Other contribution (if exists)

Appendix A – A List of GNN papers

Node Classification Part

Combining Label Propagation and Simple Models out-performs Graph Neural Networks
- Paper link: https://arxiv.org/abs/2010.13993
- Source code: https://github.com/CUAI/CorrectAndSmooth

SIGN: Scalable Inception Graph Neural Networks.
- Paper link: https://arxiv.org/abs/2004.11198
- Source code: https://github.com/twitter-research/sign

Representation learning on Graphs with Jumping Knowledge Networks:
- Paper link: https://arxiv.org/abs/1806.03536

- Source code: https://github.com/mori97/JKNet-dgl

DropEdge: Towards Deep Graph Convolutional Networks on Node Classification.
- Paper link: https://arxiv.org/abs/1907.10903
- Source code : https://github.com/DropEdge/DropEdge

ClusterGCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks
- Paper link: https://arxiv.org/abs/1905.07953
- Source code: https://github.com/google-research/google-research/tree/master/cluster_gcn

Predict then Propagate: Graph Neural Networks meet Personalized PageRank
- Paper link: https://arxiv.org/abs/1810.05997
- Source code: https://github.com/klicperajo/ppnp

Diffusion Improves Graph Learning
- Paper link: https://arxiv.org/abs/1911.05485
- Source code: https://github.com/klicperajo/gdc

GNNExplainer: Generating Explanations for Graph Neural Networks
- Paper link: https://arxiv.org/abs/1903.03894
- Source code: https://github.com/RexYing/gnn-model-explainer

PairNorm: Tackling Oversmoothing in GNNs
- Paper link: https://arxiv.org/abs/1909.12223
- Source code: https://github.com/LingxiaoShawn/PairNorm

Scalable Graph Neural Networks via Bidirectional Propagation
- Paper link: https://arxiv.org/pdf/2010.15421.pdf
- Source code: https://github.com/chennnM/GBP

Scaling Graph Neural Networks with Approximate PageRank
- Paper link: https://arxiv.org/abs/2007.01570
- Source code: https://github.com/TUM-DAML/pprgo_pytorch

Graph Classification Part

Self-Attention Graph Pooling
- Paper link: https://arxiv.org/abs/1904.08082
- Source code : https://github.com/inyeoplee77/SAGPool

StructPool: Structured Graph Pooing via Conditional Random Fields.
- Paper link: https://openreview.net/forum?id=BJxg_hVtwH
- Source code: https://github.com/Nate1874/StructPool

Convolutional Kernel Networks for Graph-Structured Data
- Paper link: https://arxiv.org/abs/2003.05189

- Source code: https://github.com/claying/GCKN

Heterogenous Graph Part

Graph Transformer for Graph-to-Sequence Learning

- Paper link: https://arxiv.org/abs/1911.07470
- Source code: https://github.com/jcyk/gtos

Heterogenous Graph Transformer

- Paper link: https://arxiv.org/abs/2003.01332
- Source code: https://github.com/acbull/pyHGT

Appendix B – Another List of GNN papers

Node Classification Part

Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering

- Paper link: https://arxiv.org/abs/1606.09375
- Source code: https://github.com/mdeff/cnn_graph
- Code in CogDL: cogdl/models/nn/pyg_cheb.py

Graph U-Nets

- Paper link: https://arxiv.org/abs/1905.05178
- Source code: https://github.com/HongyangGao/Graph-U-Nets
- Code in CogDL: cogdl/models/nn/pyg_unet.py

Dynamic Graph CNN for Learning on Point Clouds

- Paper link: https://arxiv.org/abs/1801.07829
- Source code: https://github.com/AnTao97/dgcnn.pytorch
- Code in CogDL: cogdl/models/nn/pyg_dgcnn.py

InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization

- Paper link: https://arxiv.org/abs/1908.01000
- Source code: https://github.com/fanyun-sun/InfoGraph
- Code in CogDL: cogdl/models/nn/pyg_infograph.py

Heterogenous Graph Part

Heterogeneous Graph Attention Network

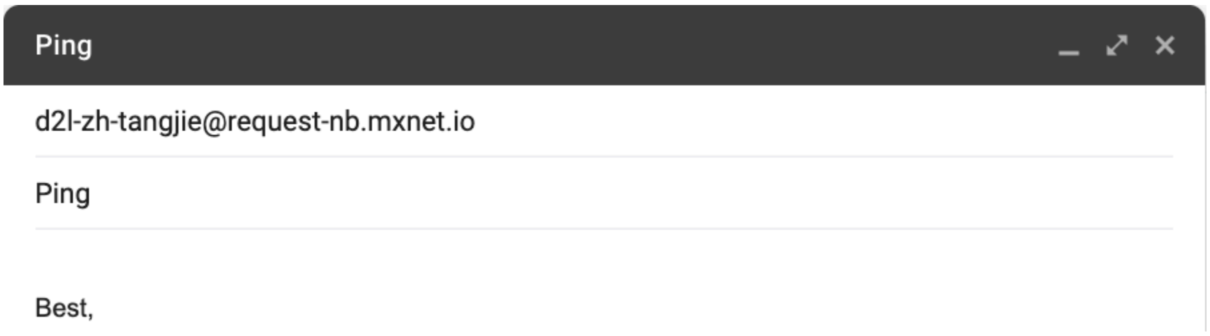
- Paper link: https://arxiv.org/abs/1903.07293
- Source code: https://github.com/Jhy1993/HAN
- Code in CogDL: cogdl/models/nn/pyg_han.py

Graph Transformer Networks

- Paper link: https://arxiv.org/abs/1911.06455
- Source code: https://github.com/seongjunyun/Graph_Transformer_Networks
- Code in CogDL: cogdl/models/nn/pyg_gtn.py

Appendix C – GPU Quota

Step 1 - Send an empty email to d2l-zh-tangjie@request-nb.mxnet.io



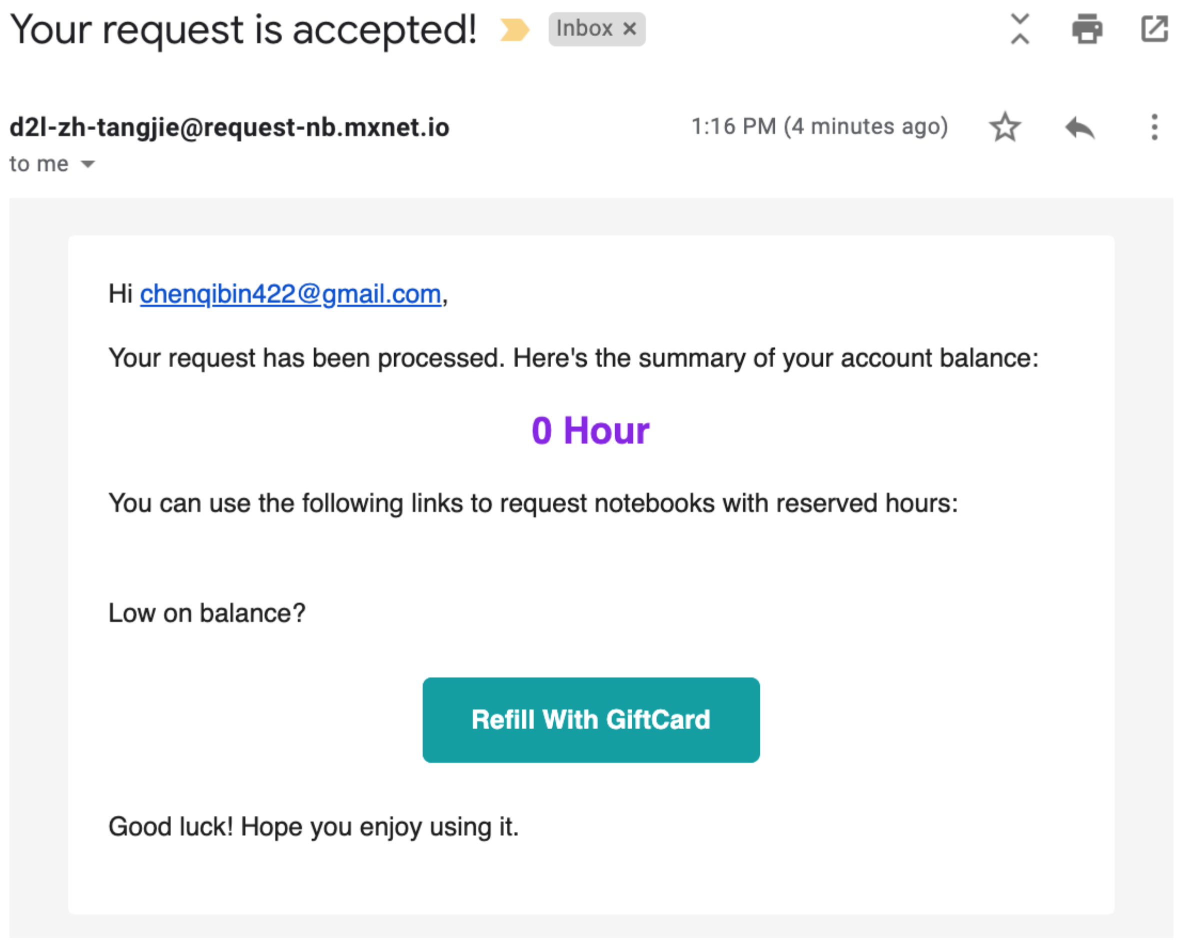
Ping

d2l-zh-tangjie@request-nb.mxnet.io

Ping

Best,

Wait for a few seconds and get an automatic reply. **Click Refill With GiftCard.**



Your request is accepted!

d2l-zh-tangjie@request-nb.mxnet.io

to me

1:16 PM (4 minutes ago)

Hi chenqibin422@gmail.com,

Your request has been processed. Here's the summary of your account balance:

0 Hour

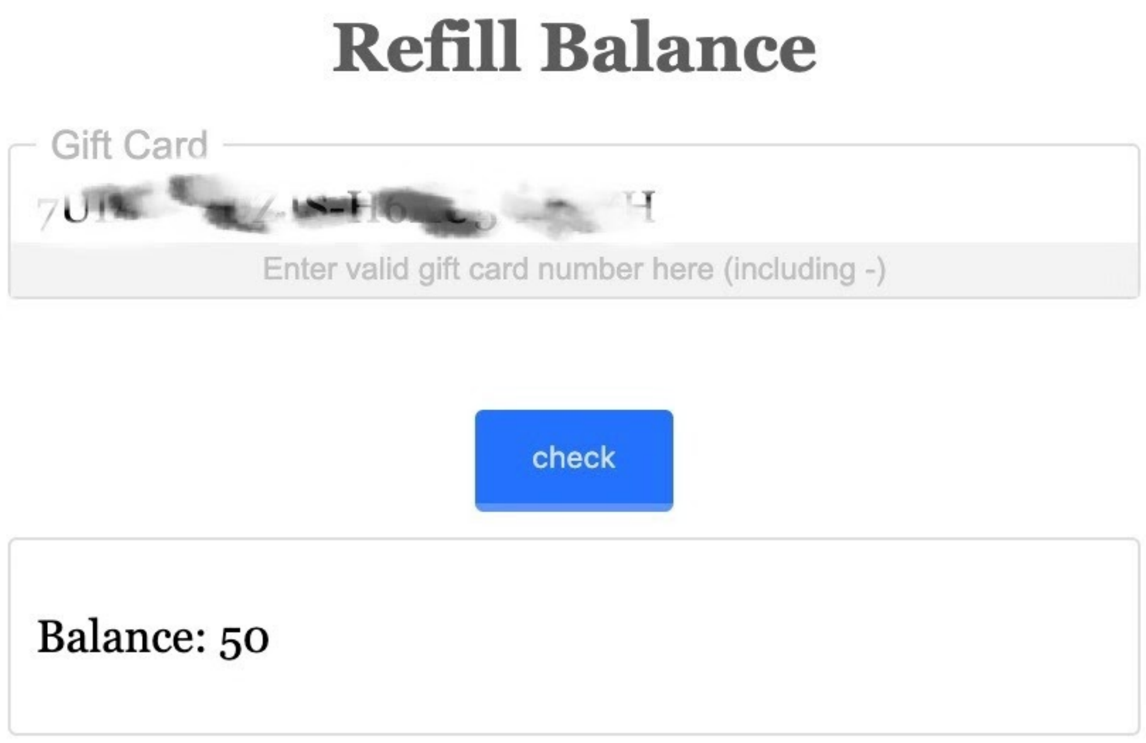
You can use the following links to request notebooks with reserved hours:

Low on balance?

Good luck! Hope you enjoy using it.

Step 2 - Refill Balance

In the pop-up page, Enter your Gift Card number and email and click **Apply**.



Refill Balance

Gift Card

[redacted]

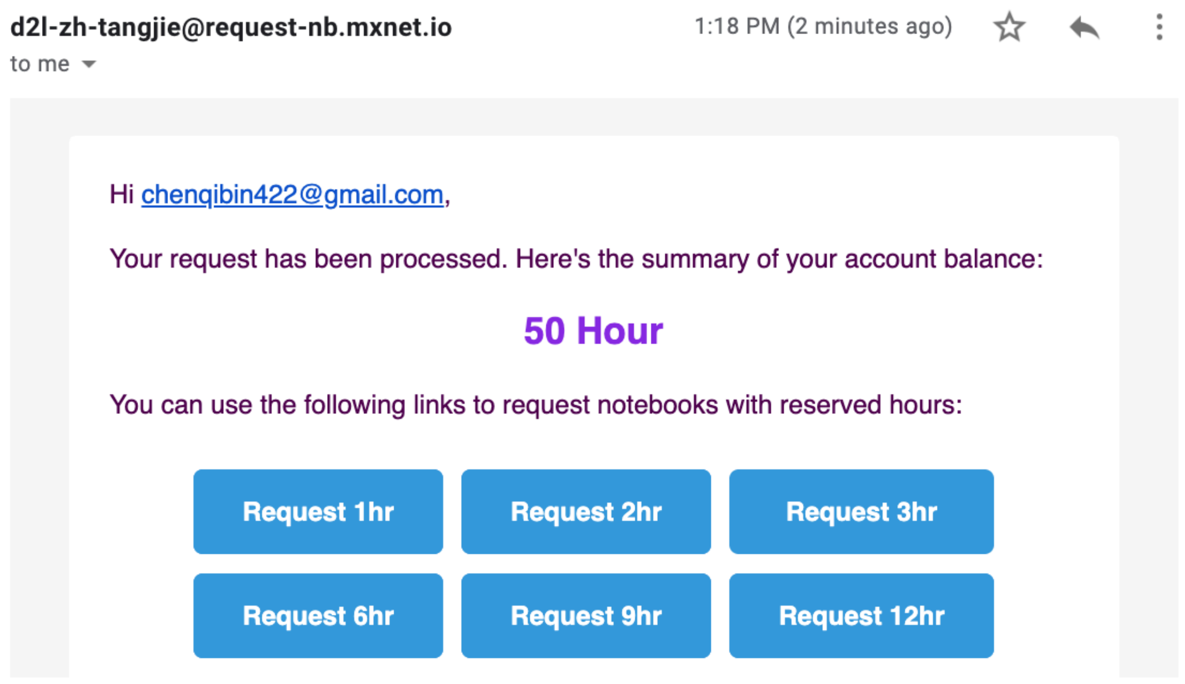
Enter valid gift card number here (including -)

Balance: 50

After successfully applying the gift card, close the page and return to your email app.

Step 3 - Send yet another empty email to d2l-zh-tangjie@request-nb.mxnet.io

Wait for the following response. Click "Request x hr" to set up an instance.



d2l-zh-tangjie@request-nb.mxnet.io

to me

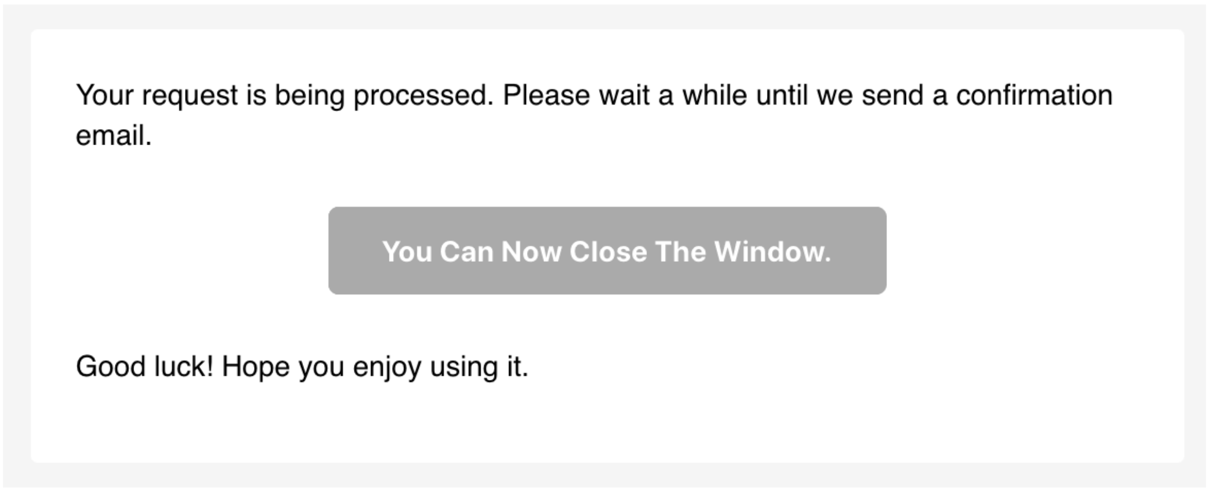
1:18 PM (2 minutes ago)

Hi chenqibin422@gmail.com,

Your request has been processed. Here's the summary of your account balance:

50 Hour

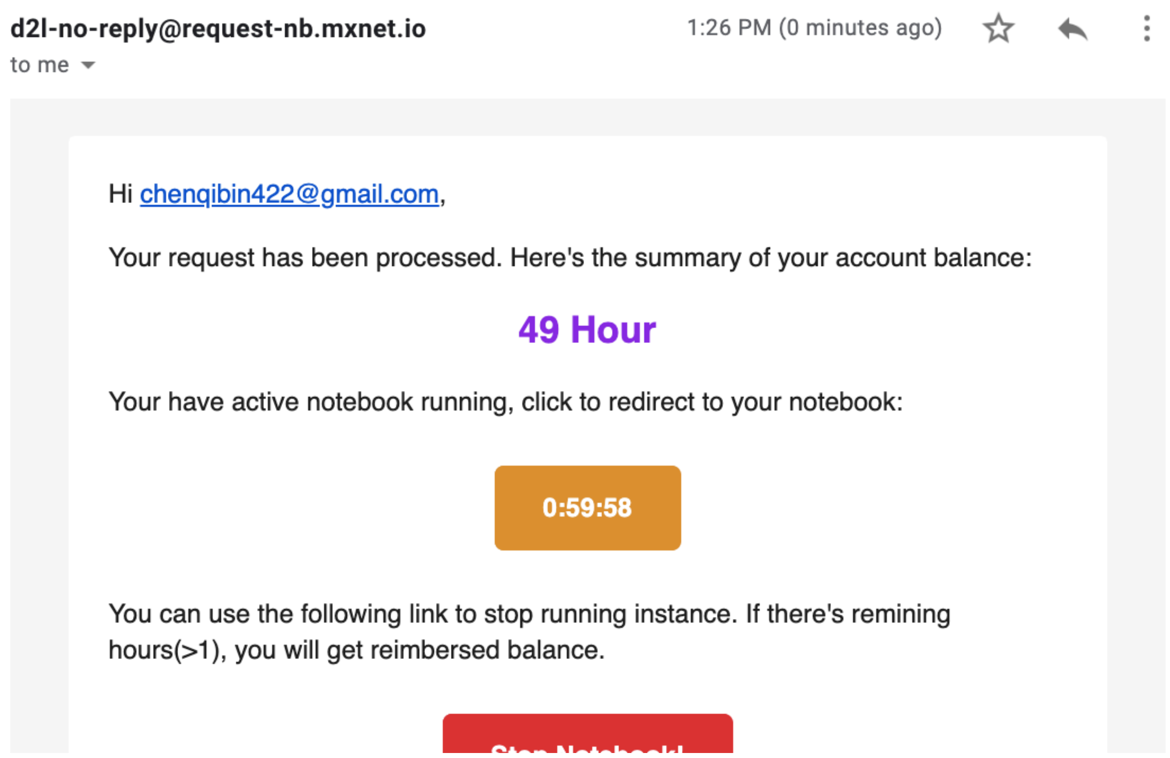
You can use the following links to request notebooks with reserved hours:


Your request is being processed. Please wait a while until we send a confirmation email.

Good luck! Hope you enjoy using it.

Step 4 - Connect to your instance

Click the **orange** button displaying the remaining time of this instance.


d2l-no-reply@request-nb.mxnet.io
to me
1:26 PM (0 minutes ago)

Hi chenqibin422@gmail.com,

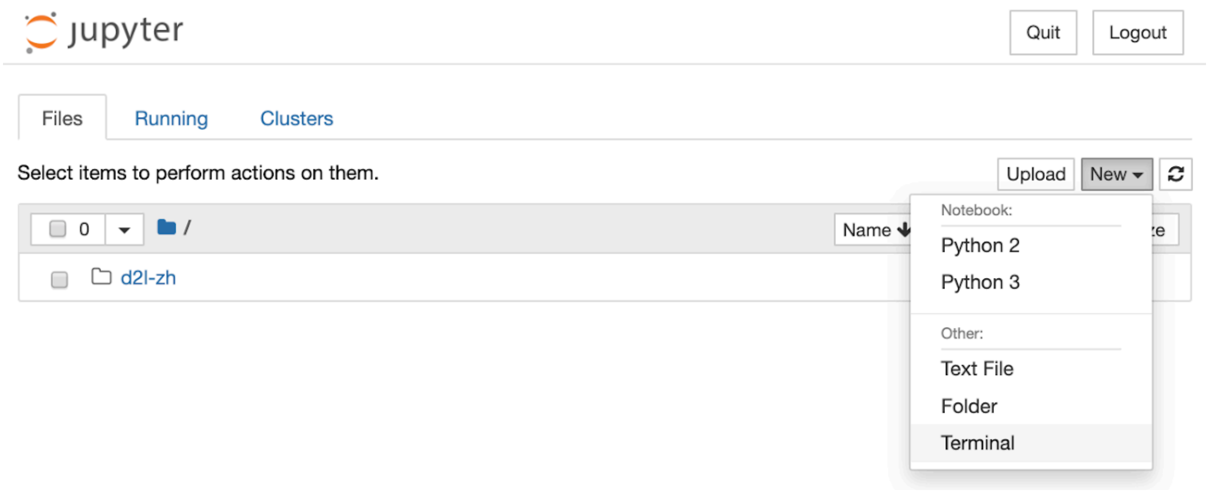
Your request has been processed. Here's the summary of your account balance:

49 Hour

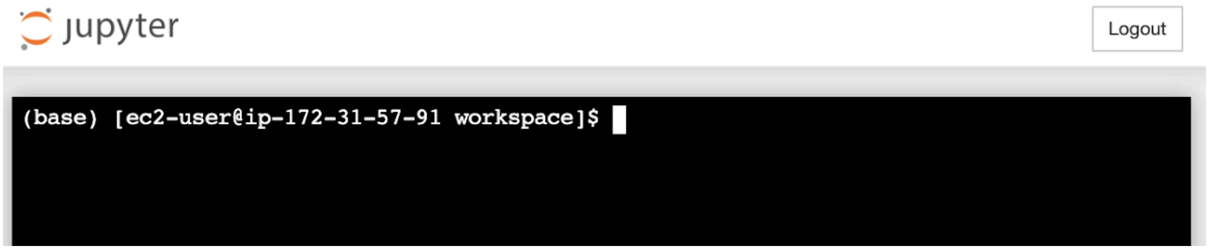
Your have active notebook running, click to redirect to your notebook:

You can use the following link to stop running instance. If there's remining hours(>1), you will get reimbersed balance.

Done! Coding time now.

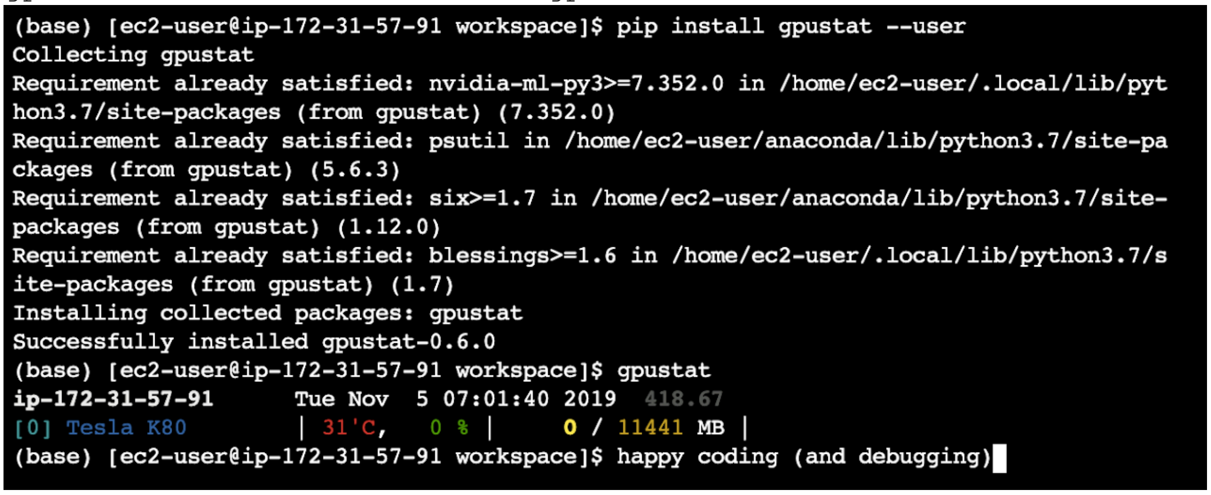


Oh wait. At least check out the **GPU**. To do this, let's start a **Terminal**.



```
(base) [ec2-user@ip-172-31-57-91 workspace]$
```

I personally prefer using `gpustat` to check out the GPUs. Type in `pip install gpustat --user` to install it and run `gpustat`.



```
(base) [ec2-user@ip-172-31-57-91 workspace]$ pip install gpustat --user
Collecting gpustat
Requirement already satisfied: nvidia-ml-py3>=7.352.0 in /home/ec2-user/.local/lib/pyt
hon3.7/site-packages (from gpustat) (7.352.0)
Requirement already satisfied: psutil in /home/ec2-user/anaconda/lib/python3.7/site-pa
ckages (from gpustat) (5.6.3)
Requirement already satisfied: six>=1.7 in /home/ec2-user/anaconda/lib/python3.7/site-
packages (from gpustat) (1.12.0)
Requirement already satisfied: blessings>=1.6 in /home/ec2-user/.local/lib/python3.7/s
ite-packages (from gpustat) (1.7)
Installing collected packages: gpustat
Successfully installed gpustat-0.6.0
(base) [ec2-user@ip-172-31-57-91 workspace]$ gpustat
ip-172-31-57-91      Tue Nov  5 07:01:40 2019  418.67
[0] Tesla K80         | 31'C,   0 % |      0 / 11441 MB |
(base) [ec2-user@ip-172-31-57-91 workspace]$ happy coding (and debugging)
```

It's a **Tesla K80**.

One more thing

Should I worry about losing all my uploaded files when the instance time is up?

By far I've tried two 1hr instances and found they share the storage, so most probably you can still access your data until your whole 100hr is used up. However, it's good practice to backup your data.

If you have any questions or suggestions, please feel free to send an email to me cyk20@mails.tsinghua.edu.cn.

Step 5 (optional) - Jupyter Notebook

In case you are not familiar with Jupyter, check out this great introduction article. Although the interactive environment is easy to use for beginners, it's not convenient when dealing with multiple code files. We recommend to code on local machine with your favorite IDE and upload your code to the instance and run it in Terminal.