Movie Recommendations

Group Project Presentation

[Machine Learning in Production]

Team: K-Avengers

Yoonseok Heo, Yein Park,
Hogeon Yu, Dain Kim, Minwoo Choi

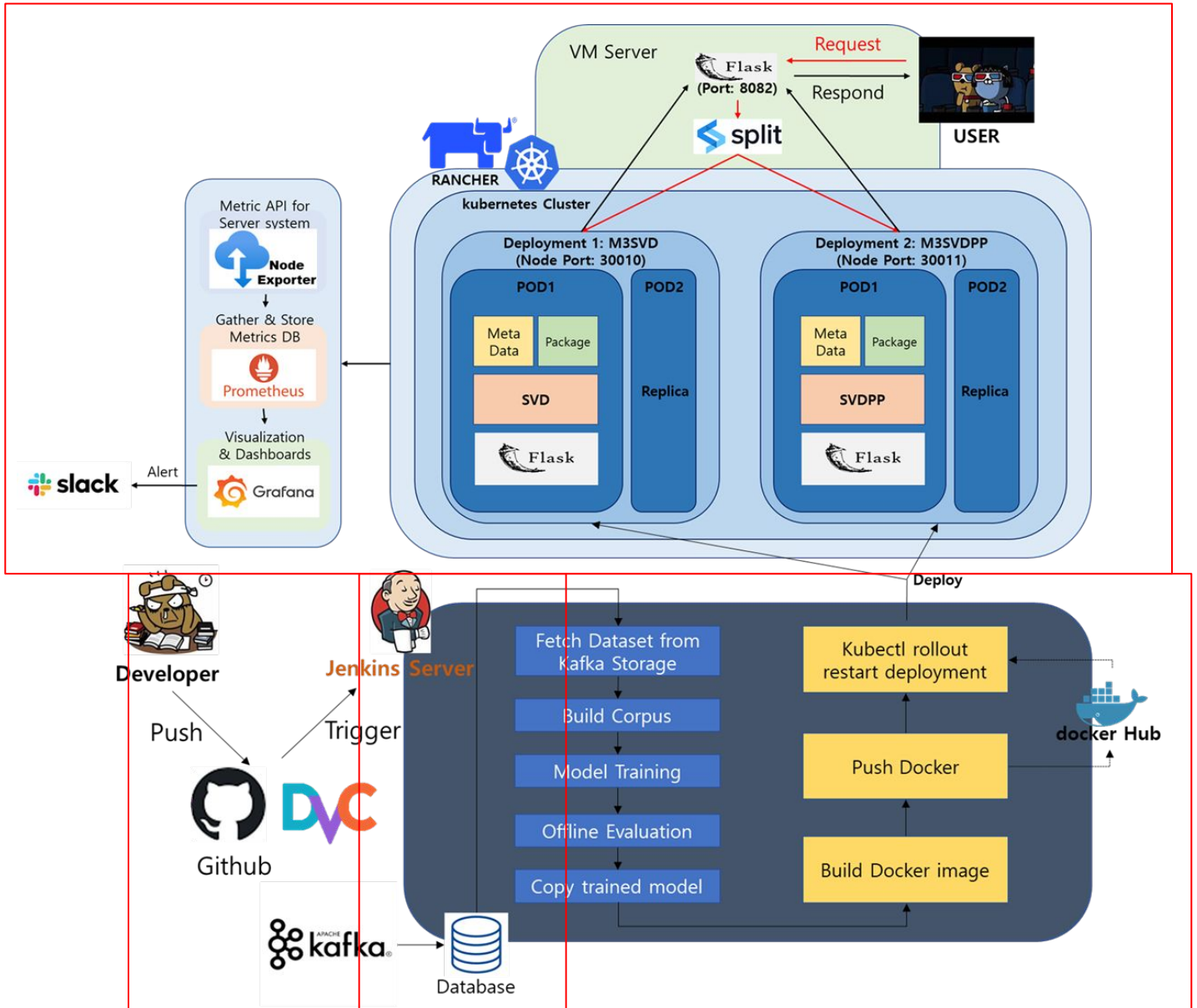
Carnegie Mellon University

Contents

Carnegie Mellon University

- Introduction (20s)
- Overall Architecture (40s)
- Continuous Integration
  - A pipeline for movie recommendation (40s)
  - Code integrity checks with unit-test (40s)
  - Automatic integration pipeline with Jenkins (1m)
- Continuous Deployment
  - Containerization with Rancher (30s)
  - Automatic Continuous Deployment with Jenkins(Contribution) (1m30s)
  - Monitoring (40s)
  - Versioning and tracking provenance(40s)
- Reflection(1m)
- Conclusion(20s)

Overall Architecture



Contents

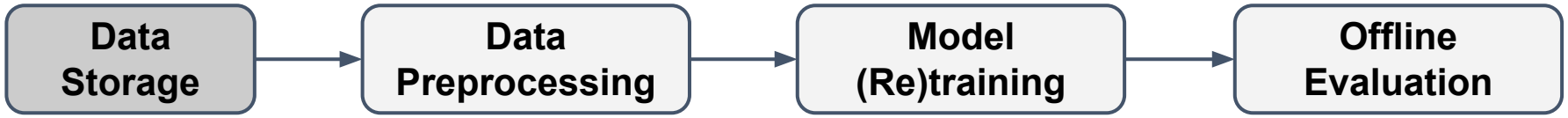
Carnegie Mellon University

- Introduction (20s)
- Overall Architecture (40s)
- **Continuous Integration**
  - A pipeline for movie recommendation (40s)
  - Code integrity checks with unit-test (40s)
  - Automatic integration pipeline with Jenkins (1m)
- Continuous Deployment
  - Containerization with Rancher(30s)
  - Automatic Continuous Deployment with Jenkins(Contribution)(1m30s)
  - Monitoring (40s)
  - Versioning and tracking provenance(40s)
- Reflection(1m)
- Conclusion(20s)

A pipeline for movie recommendation

Carnegie Mellon University

- **Overall CI pipeline for movie recommendation**

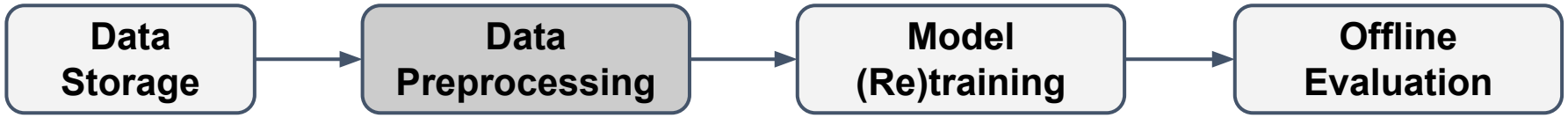


- **Data storage**
  - **Apache Kafka** is a distributed event store and stream-processing platform.
  - Collect Kafka log data
    - Data (movies watched by user) → for (re)training model and for online evaluation
    - Rate (rating by user) → for (re)training model and for online evaluation
    - Request → for online evaluation
  - This pipeline, once run, continues to run until it is intentionally stopped.
  - After online evaluation, expired data is automatically deleted.

A pipeline for movie recommendation

Carnegie Mellon University

- **Overall CI pipeline for movie recommendation**

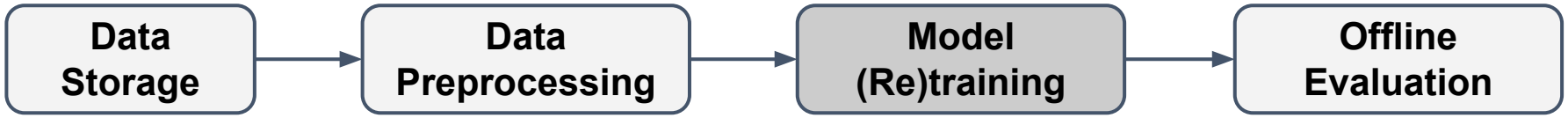


- **Data preprocessing**
  - Pre-process the stored raw data
  - Generate a compressed sparse row (CSR) matrix
  - Split it into train/validation sets

A pipeline for movie recommendation

Carnegie Mellon University

- **Overall CI pipeline for movie recommendation**



- **Model (re)training**
  - Matrix Factorization (MF)
    - SVD

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

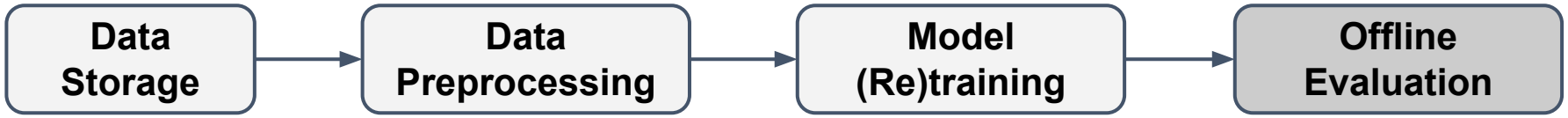
    - SVD++

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j \right)$$

A pipeline for movie recommendation

Carnegie Mellon University

- **Overall CI pipeline for movie recommendation**



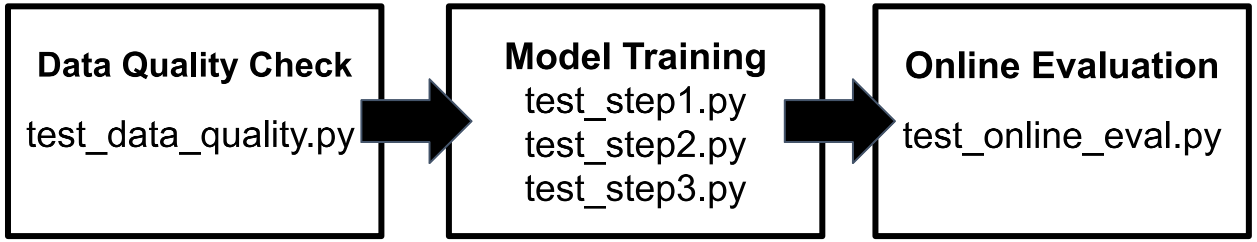
- **Offline evaluation**
  - We use 'RMSE' as metric for offline evaluation.
  - Root mean square error (RMSE)
    - $$RMSE = \sqrt{\frac{\sum_{(i,j)\in E} (r_{ij} - \widehat{r_{ij}})^2}{|E|}}, \quad where\ E = nonzero\ entries\ in\ the\ testset.$$

Code Integrity Check with Unit-test

Carnegie Mellon University

- **Unit-test**
  - Unittest
    - Python’s built in library.
  - The process is integrated on Jenkins pipeline, which runs automatically.
  - The result can be identified in a coverage report format on Jenkins.

<Unit–test Pipeline>



<Unit–test Coverage Report>

```
+ coverage report
Name                                                                                              Stmts   Miss  Cover
---------------------------------------------------------------------------------------------------------------------
/home/team24/milestone2/group-project-s22-k-avengers/Milestone2/model_pipeline/Data_Management/Process_RawData.py      36     18    50%
/home/team24/milestone2/group-project-s22-k-avengers/Milestone2/model_pipeline/Data_Management/__init__.py             0      0   100%
/home/team24/milestone2/group-project-s22-k-avengers/Milestone2/model_pipeline/For_Jenkins/test_step1.py              12      0   100%
/home/team24/milestone2/group-project-s22-k-avengers/Milestone2/model_pipeline/path/__init__.py                        0      0   100%
/home/team24/milestone2/group-project-s22-k-avengers/Milestone2/model_pipeline/path/path.py                            3      0   100%
---------------------------------------------------------------------------------------------------------------------
TOTAL                                                                                                51     18    65%
+ coverage html
Wrote HTML report to htmlcov/index.html
+ echo Finished coverage report
Finished coverage report
```

Automatic integration pipeline with Jenkins

Carnegie Mellon University

- **Continuous integration**
  - Jenkins
    - Pipeline
      - Unit test 1 to 5 → model management & offline evaluation(model) → online evaluation
    - Using Blue Ocean plugin
      - A more visualized dashboard than ever before
      - Commit occurs in master branch of github → Autorun the entire pipeline
      - Save after pipeline build → Jenkinsfile for pipeline is committed to master branch on github
    - Using freestyle project
      - Automatically run once in a specific period of time
      - Setting the “build periodically” option

```
pipeline {
  agent any
  stages {
    stage('unit-1') {
      steps {
        sh '''pwd
pip3 install matplotlib
pip3 install numpy

echo \'unit 1\'

coverage run --omit=\'/usr/lib*\' /home/team24/group-project-s

coverage report

echo \'Finished coverage report\''''
      }
    }

    stage('unit-2') {
      steps {
        sh '''echo \'unit 2\'

pwd

PYTHONPATH=/home/team24/group-project-s22-k-avengers/Milestone

coverage report

```

Contents

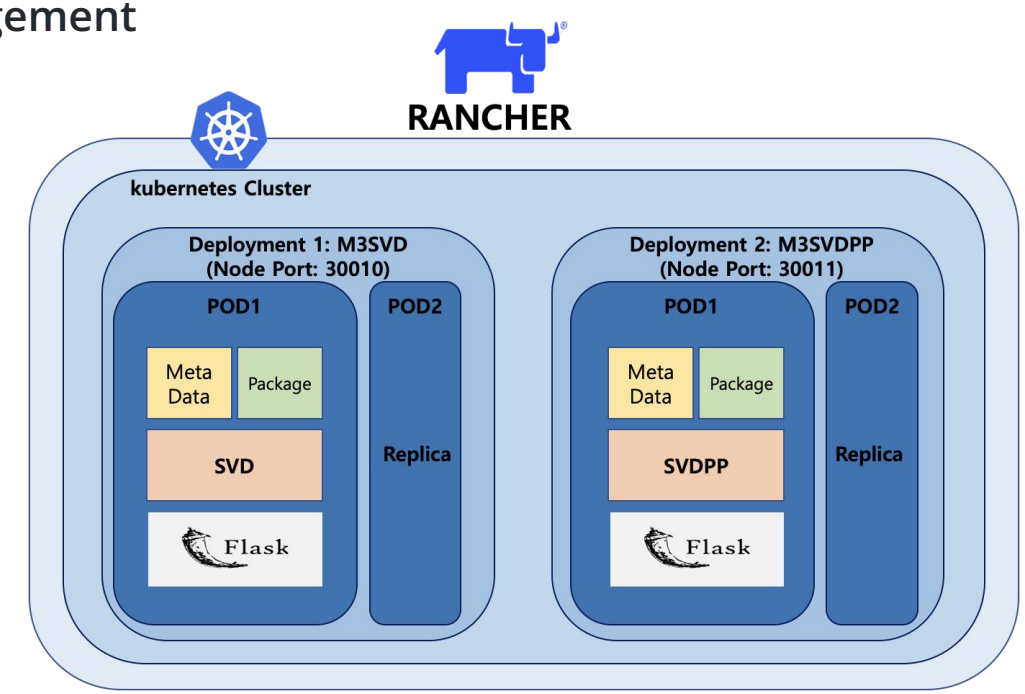
Carnegie Mellon University

- Introduction (20s)
- Overall Architecture (40s)
- Continuous Integration
  - A pipeline for movie recommendation (40s)
  - Code integrity checks with unit-test (40s)
  - Automatic integration pipeline with Jenkins (1m)
- **Continuous Deployment**
  - Containerization with Rancher (30s)
  - Automatic Continuous Deployment with Jenkins(Contribution) (1m30s)
  - Monitoring (40s)
  - Versioning and tracking provenance(40s)
- Reflection(1m)
- Conclusion(20s)

Containerization with Rancher

Carnegie Mellon University

- **Containerization**
  - Rancher
    - A complete container management platform that includes everything necessary for container management during the production process
  - Deployment components
    - Our system manages two recommendation models as different deployments in one cluster
    - Each deployment consists of two **pods**, **one replica of the other**, which distributes and processes tasks



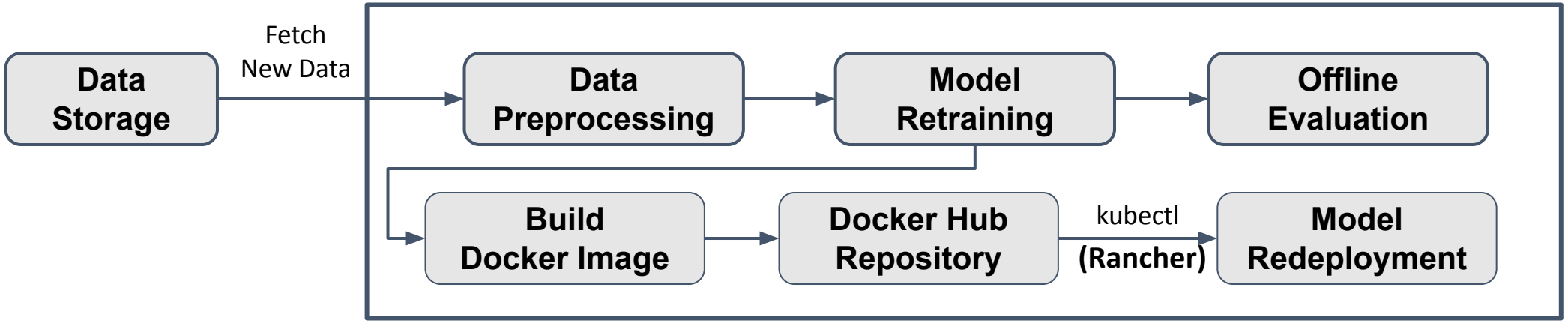
Overview of Our Deployments

Automatic Continuous Deployment with Jenkins

Carnegie Mellon University

- **Contribution: Automatic Continuous Deployment with Jenkins**
  - Extending our integration pipeline to model deployment
  - We leverage jenkins to transmit the deployment signal to the Rancher
  - Whenever committed to Github, the pipeline is executed:
    - Continuous Integration: Data fetching, Data preprocessing, Model retraining
    - **Continuous Deployment**: Build docker images, Push images to docker repo
    - **Model deployment**: Pull docker images for retrained models and redeploy it through Rancher

(* command: kubetcl rollout restart deployment)



CI + CD Pipeline with Jenkins

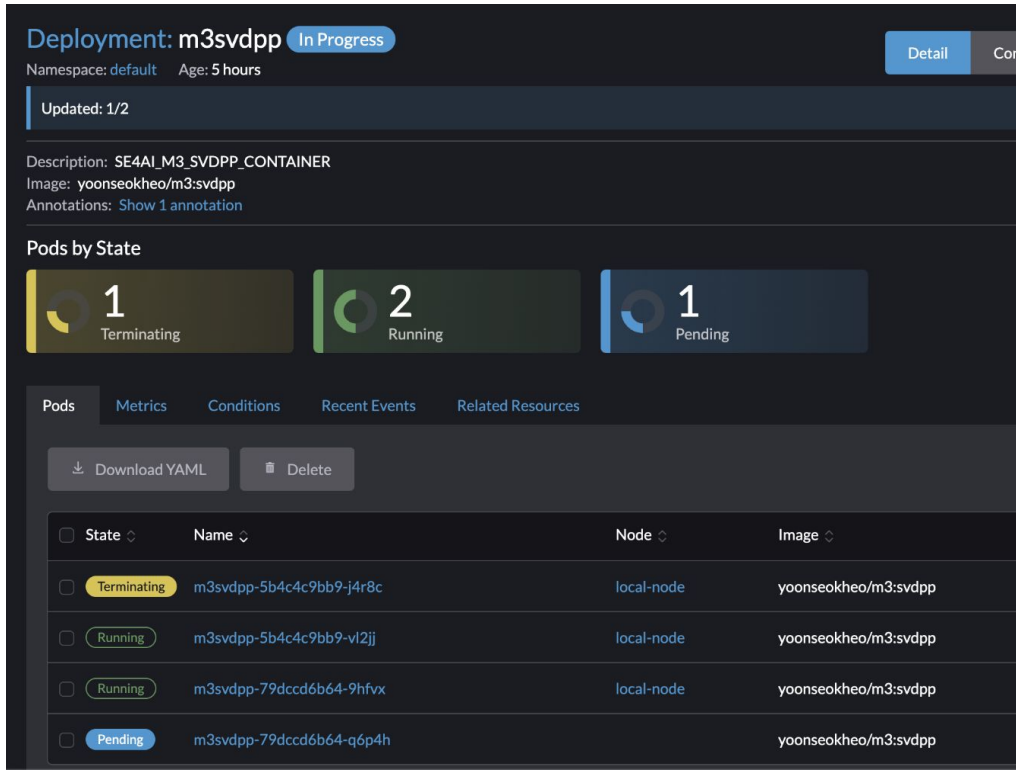
Model Re-deployment without Downtime

Carnegie Mellon University

- **Contribution: ZERO downtime for model redeployment**
  - The new redeployment also has 2 pods with replica
  - After one new pod is deployed, one existing pod is terminated
  - After a new pod is deployed again,
    the remaining existing pod is also terminated

  ⇒ **ZERO downtime** in the process of deploying the re-trained models

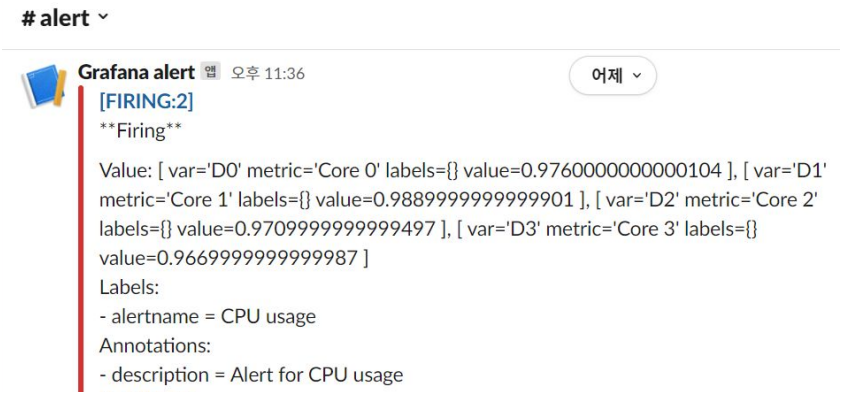
- All these processes are stably controlled under the Rancher platform, and even now, the process of retraining and redeployment is being stably operated periodically

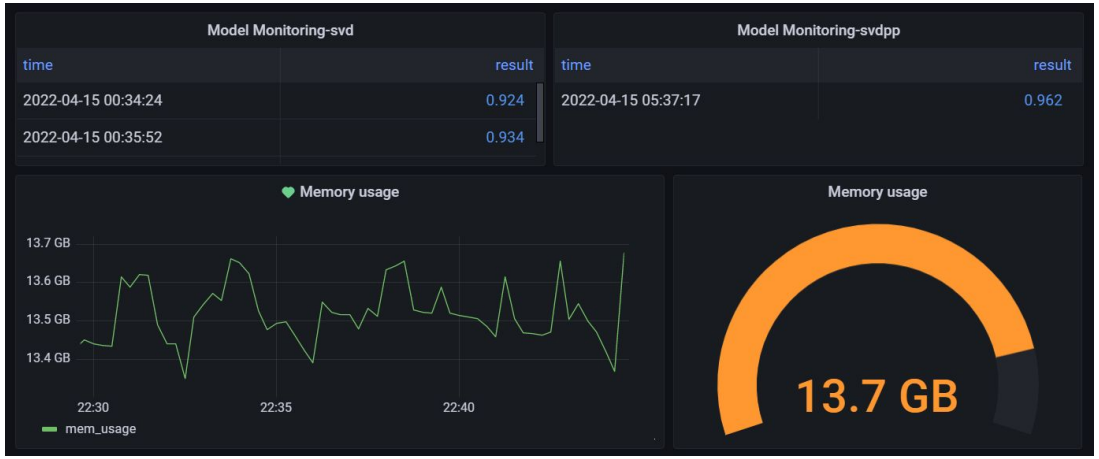


Example: Redeployment process with ZERO downtime

Monitoring

- **Monitoring infrastructure**
  - Prometheus, Grafana and Node Exporter to monitor our infrastructure
    - Memory usage
    - CPU usage
    - Latency time in flask
    - Model quality
  - Sending alerts to our slack #alert channel

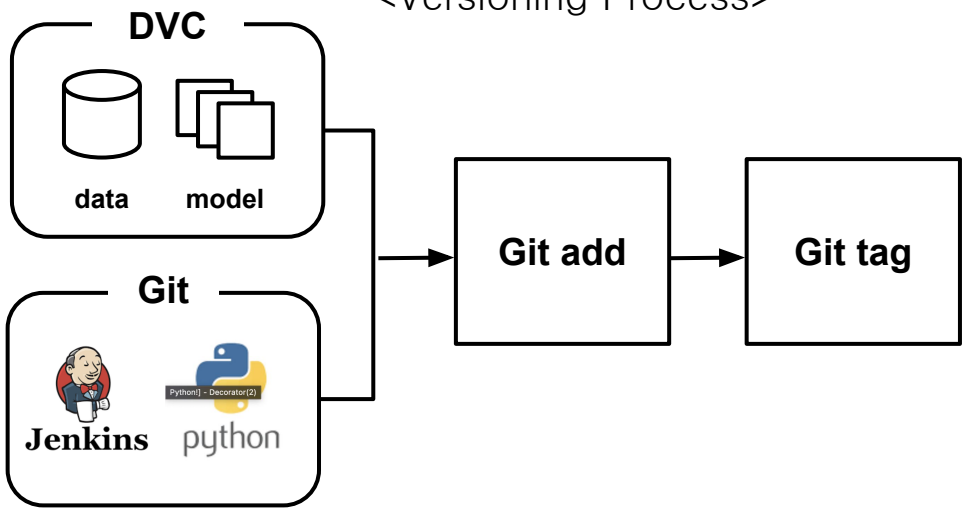




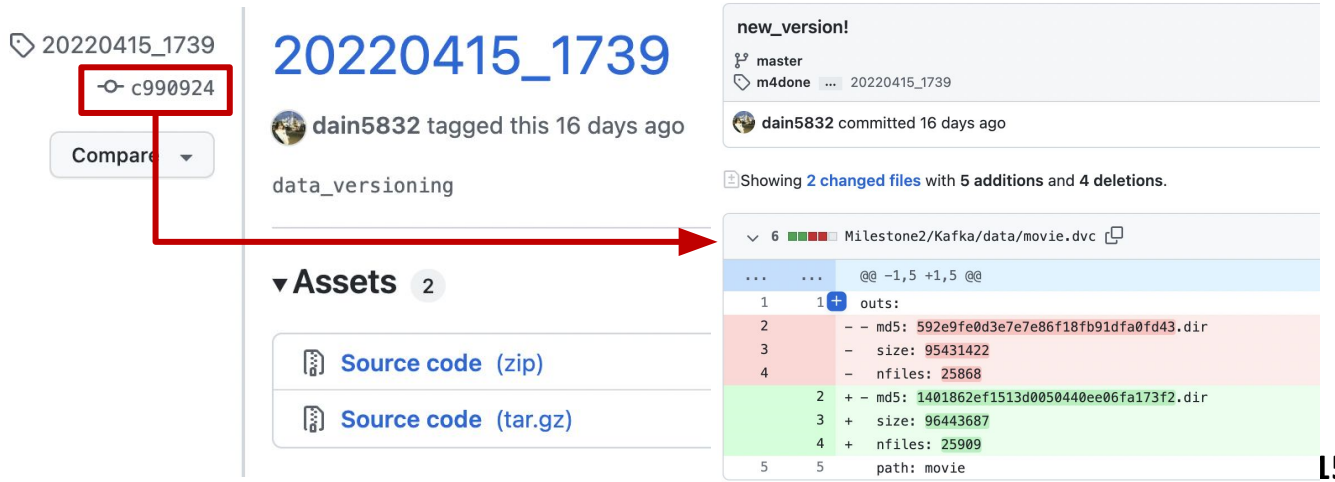
Versioning and Tracking Provenance

- **Provenance**
  - DVC
    - an open-source version control system.
    - DVC stores the information of dataset and the model in .dvc format.
  - Process
    - track modification → add changes to git → push git tag

<Versioning Process>



<Version Tracking Example>



Reflection

Carnegie Mellon University

- Things went very well 
  - CI & CD
    - One pipeline using Jenkins platform
  - Zero-downtime
    - Rolling update with replica in Rancher platform
  - A/B test
    - Continuous A/B test & Load balancing
- Things went less well 
  - Less well used Git branch
  - Misallocated storage
  - Less efficiently written code
  - Server maintenance issue

Conclusion

Carnegie Mellon University

We have learned how to:

- Collect data from multiple sources and engineer features for learning
- Deploy and measure a model inference service
- Build and operate infrastructures
  - a continuous integration infrastructure for evaluate a model in production
  - a monitoring infrastructure for system health and model quality
  - a continuous deployment infrastructure for automatic periodic retraining and versioning
- Design and implement a monitoring strategy to detect possible issues in machine-learning systems.

We practiced teamwork and had a fun!!!

Q&A

Carnegie Mellon University