

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

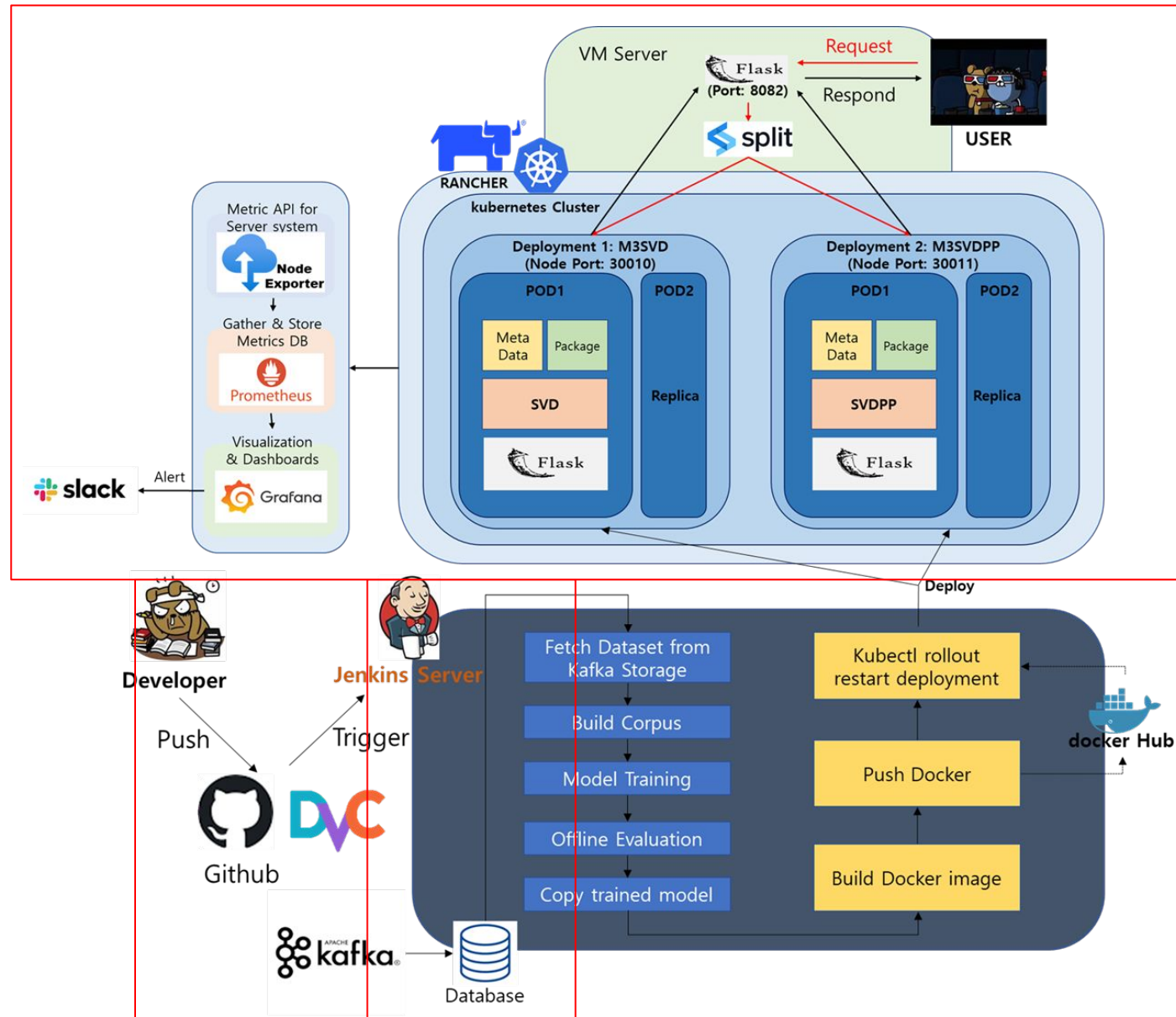


Contents

- 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

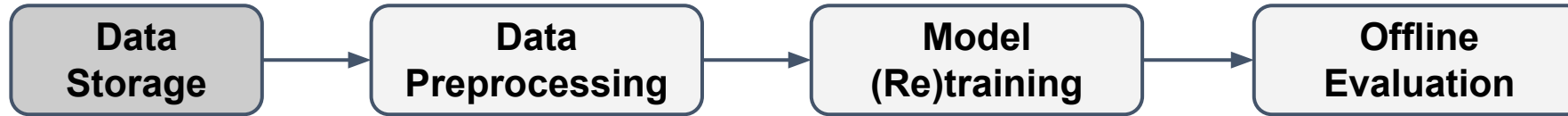


Contents

- 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

- Overall CI pipeline for movie recommendation

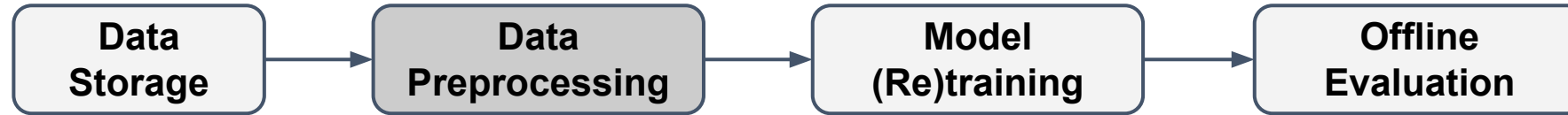


- Data storage

- 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

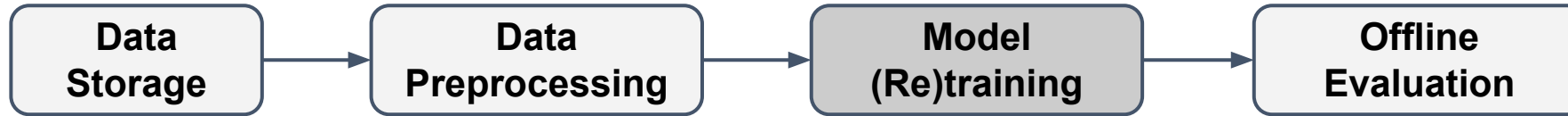
- Overall CI pipeline for movie recommendation



- 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

- Overall CI pipeline for movie recommendation



- Model (re)training

- Matrix Factorization (MF)

- SVD

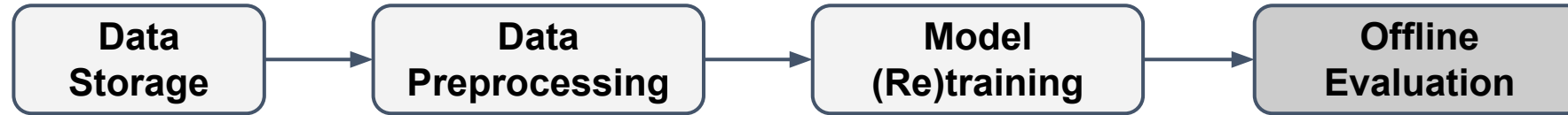
$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

- 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

- Overall CI pipeline for movie recommendation



- Offline evaluation

- We use 'RMSE' as metric for offline evaluation.
- Root mean square error (RMSE)

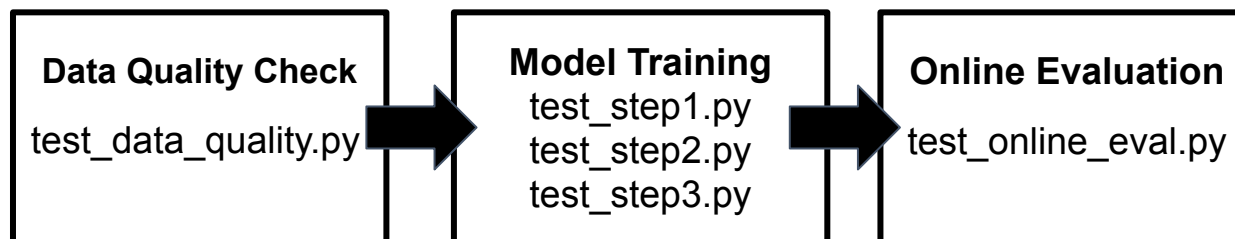
■

$$RMSE = \sqrt{\frac{\sum_{(i,j) \in E} (r_{ij} - \hat{r}_{ij})^2}{|E|}}, \text{ where } E = \text{nonzero entries in the testset.}$$

Code Integrity Check with Unit-test

- **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
-----
/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

- 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

```
1 pipeline {
2   agent any
3   stages {
4     stage('unit-1') {
5       steps {
6         sh '''pwd
7         pip3 install matplotlib
8         pip3 install numpy
9
10        echo \'unit 1\'
11
12        coverage run --omit=\'/usr/lib*\' /home/team24/group-project-:
13
14        coverage report
15
16        echo \'Finished coverage report\'
17      }
18    }
19
20    stage('unit-2') {
21      steps {
22        sh \'\'\'echo \'unit 2\'
23
24        pwd
25
26        PYTHONPATH=/home/team24/group-project-s22-k-avengers/Milestone
27
28        coverage report
29
```



Contents

- 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

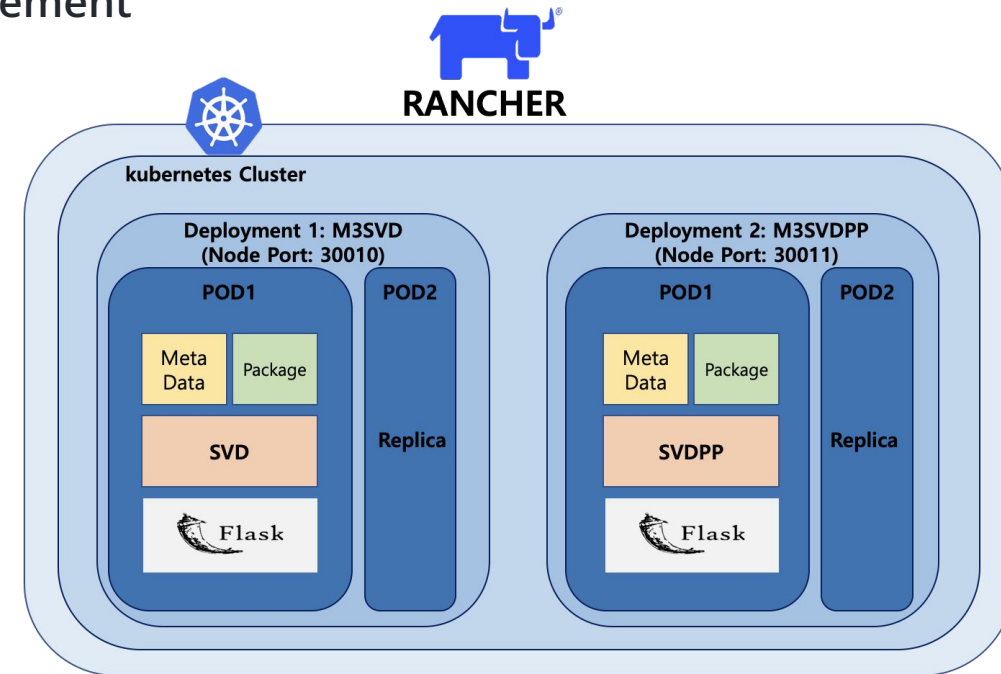
- Containerization

- 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



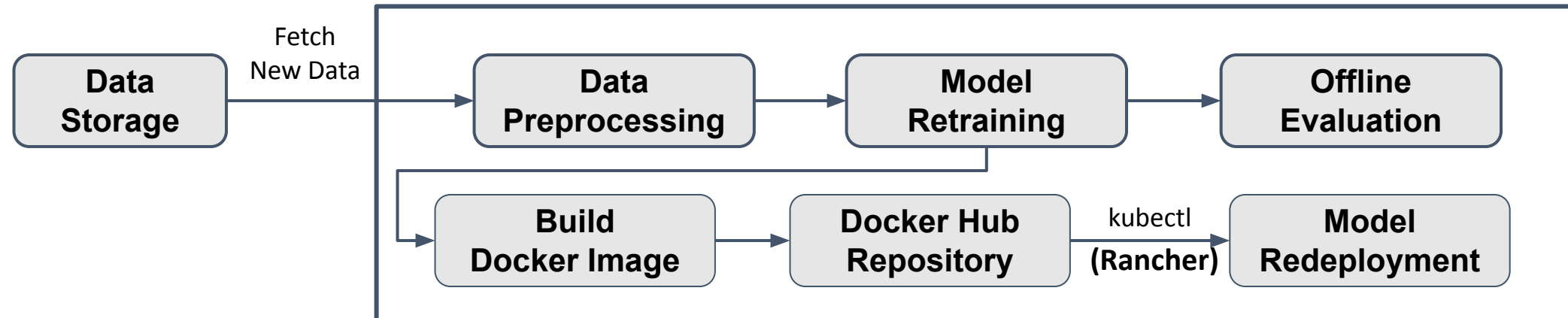
Overview of Our Deployments



Automatic Continuous Deployment with Jenkins

- **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: kubectl rollout restart deployment)



Model Re-deployment without Downtime

- **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

The screenshot displays the Rancher UI for a deployment named 'm3svdpp' in the 'default' namespace. The deployment is in an 'In Progress' state, updated 1/2. The description is 'SE4AI_M3_SVDPP_CONTAINER' using the image 'yoonseokheo/m3:svdpp'. The 'Pods by State' section shows 1 Terminating pod, 2 Running pods, and 1 Pending pod. Below this, a table lists the pods with their names, nodes, and images.

State	Name	Node	Image
Terminating	m3svdpp-5b4c4c9bb9-j4r8c	local-node	yoonseokheo/m3:svdpp
Running	m3svdpp-5b4c4c9bb9-vl2jj	local-node	yoonseokheo/m3:svdpp
Running	m3svdpp-79dccc6b64-9hfvx	local-node	yoonseokheo/m3:svdpp
Pending	m3svdpp-79dccc6b64-q6p4h		yoonseokheo/m3:svdpp

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



alert ▾



Grafana alert 오후 11:36

어제 ▾

[FIRING:2]

Firing

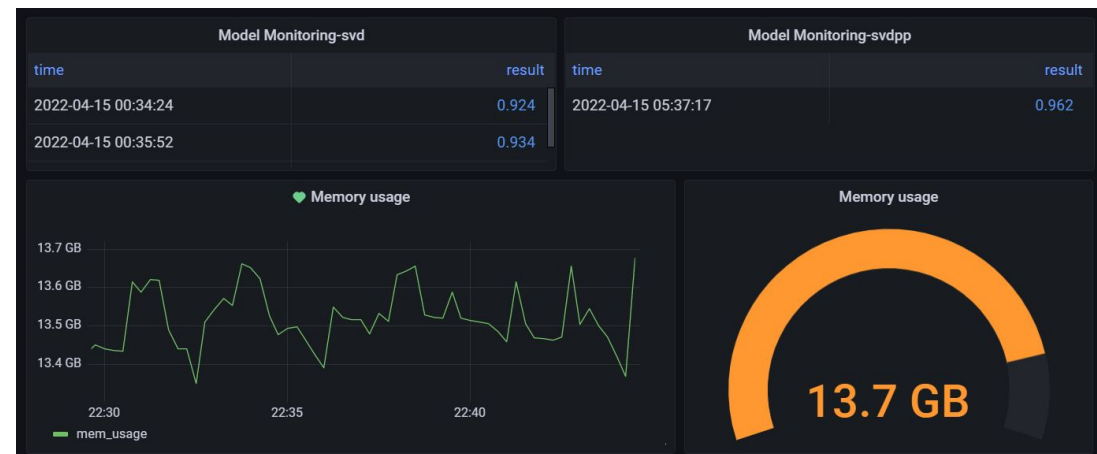
Value: [var='D0' metric='Core 0' labels={} value=0.9760000000000104], [var='D1' metric='Core 1' labels={} value=0.9889999999999901], [var='D2' metric='Core 2' labels={} value=0.97099999999999497], [var='D3' metric='Core 3' labels={} value=0.9669999999999987]

Labels:

- alertname = CPU usage

Annotations:

- description = Alert for CPU usage



Versioning and Tracking Provenance

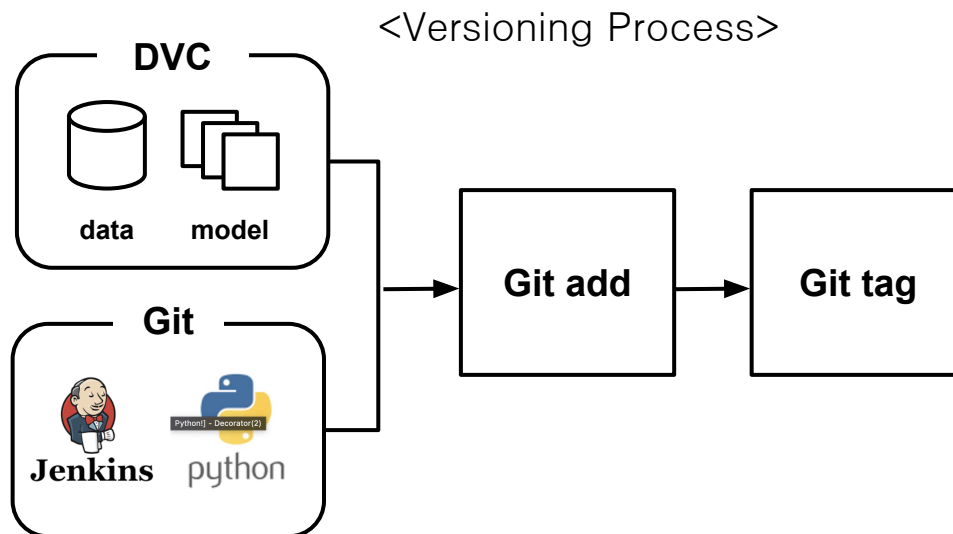
- Provenance

- 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



<Version Tracking Example>

The screenshot shows a Git commit page for the tag **20220415_1739**. The commit message is `data_versioning`. A red box highlights the commit hash `c990924` under the tag. A red arrow points from this hash to the 'Assets' section, which lists two files: `Source code (zip)` and `Source code (tar.gz)`. On the right, a 'new_version!' section shows the commit details, including the user `dain5832` and the commit message `new_version!`. Below this, a table shows the changes in the file `Milestone2/Kafka/data/movie.dvc`.

Line	Change	File
1	1 +	outs:
2	-	md5: 592e9fe0d3e7e7e86f18fb91dfa0fd43.dir
3	-	size: 95431422
4	-	nfiles: 25868
2	+	md5: 1401862ef1513d0050440ee06fa173f2.dir
3	+	size: 96443687
4	+	nfiles: 25909
5	5	path: movie

Reflection

- 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

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!!!

