MLOps:
From Jupyter to Production

Tommy Fernandez
Technical Writer at Semaphore

/TomFernBlog

Where to find the code

Here you can find all the code used during the talk.

- Repository:https://github.com/semaphoreci-demos/semaphore-demo-mlops
- Jupyter Notebook:https://www.kaggle.com/code/tomasfern/cats-or-dogs-classifier/
- Dataset:https://www.robots.ox.ac.uk/~vgg/data/pets/
- Dataset (alternative link):https://www.kaggle.com/datasets/tomasfern/oxford-iit-pets
- Tutorial Video 1:https://www.youtube.com/watch?v=OrydpKLDKuk
- Tutorial Video 2:https://www.youtube.com/watch?v=OEVuRyGK5zQ
- Blog post:https://semaphoreci.com/blog/machine-learning

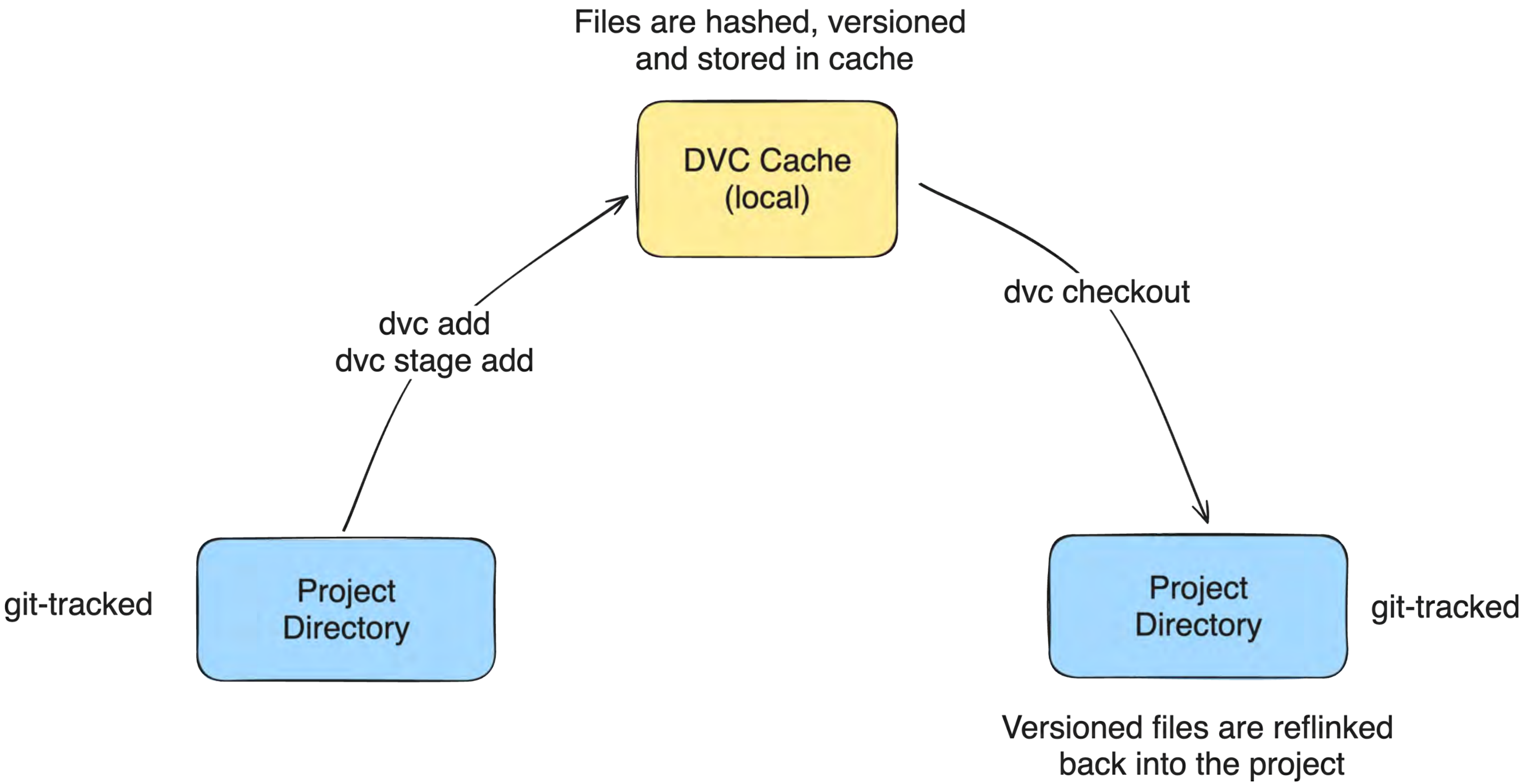
Why MLOps?

- Less work
- Scaling
- Consistency
- Traceability

Why use DVC?

- Data used in training is tracked
- Data, procedures and results can be shared
- Reproducible training
- Allows quick experimentation with parameters tracking
- Cache results to speed up training steps
- Integration with CI/CD and DVC Enterprise

How the DVC cache works



Why use ML Pipelines?

- Makefile for ML
- Versioned with Git
- Results are cached

Setting up the experiment

```
# clone repository
$ git clone https://github.com/semaphoreci-demos/semaphore-demo-mlops.git

# download dataset
$ wget https://huggingface.co/datasets/tomfern/oxford-pets-subset/resolve/main/images.tar.gz -O data/images.tar.gz

# track image.tar.gz
$ dvc add data/images.tar.gz
```

Adding a Stage

```
$ dvc stage add --name train \
                --deps src/model.py \
                --deps data/clean.csv \
                --outs data/predict.dat \
                python src/model.py data/clean.csv
```

Format for dvc.yaml

```yaml
stages:
  prepare:
    ...
    outs:
      - data/clean.csv
  train:
    cmd: python src/model.py data/model.csv
    deps:
      - src/model.py
      - data/clean.csv
    outs:
      - data/predict.dat
```

Add all stages

```
# prepare stage
$ dvc stage add -n prepare \
   -d src/prepare.py \
   -o data/images \
   python src/prepare.py

# train stage
$ dvc stage add -n train \
   -d src/train.py -d data/images \
   -o models/model.pkl -o models/model.pth \
   -m metrics/classification.md \
   --plots metrics/confusion_matrix.png \
   --plots metrics/top_losses.png \
   --plots metrics/finetune_results.png \
   python src/train.py

# test stage
$ dvc stage add -n test \
   -d src/test.py -d models/model.pkl -d models/model.pth \
   python src/test.py
```

Run all stages

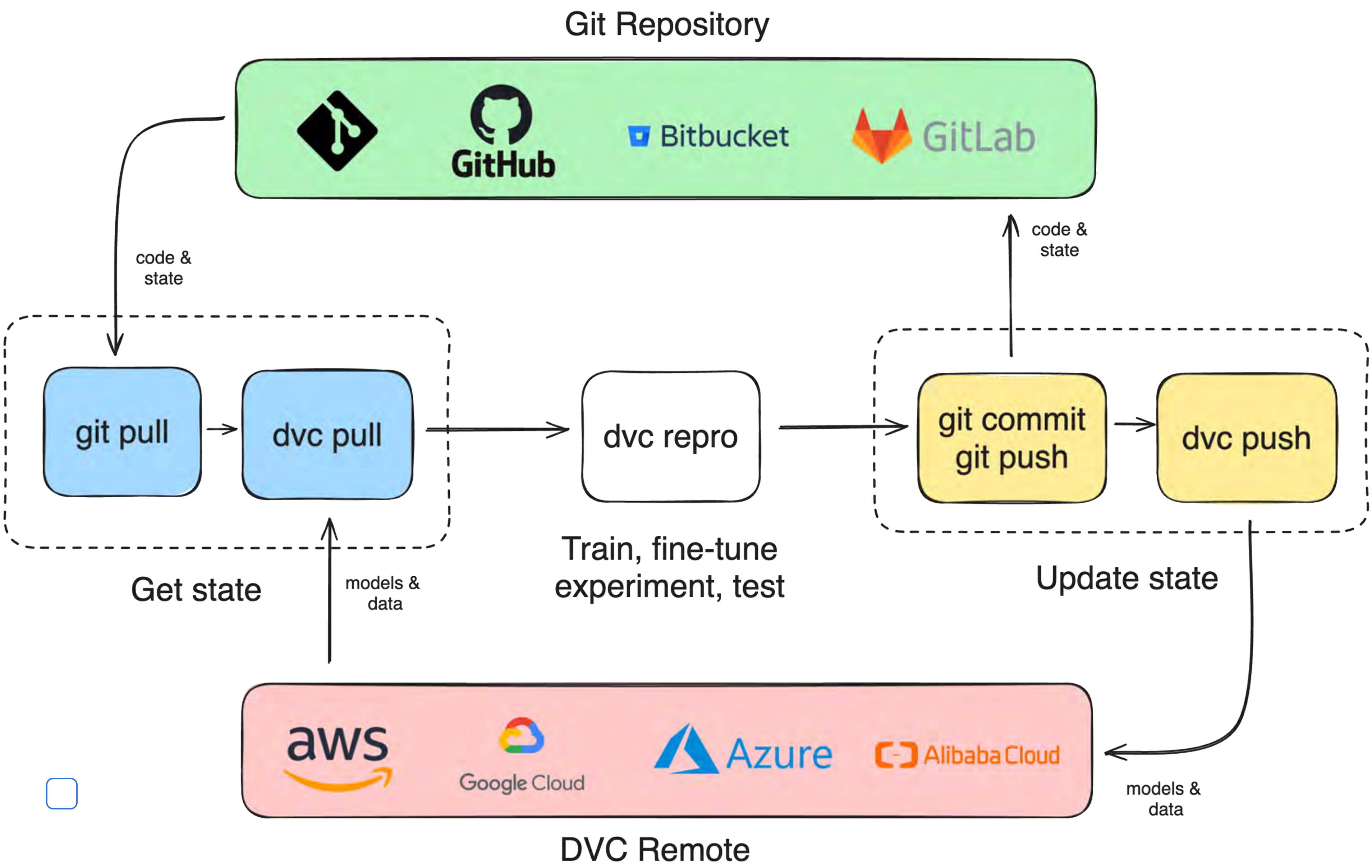
```
dvc repro

Running stage 'prepare':
> python src/prepare.py
Decompressing data/images.tar.gz...

Running stage 'train':
> python src/train.py
Image count for dataset
- Training: 1293
- Validation: 554
epoch     train_loss  valid_loss  error_rate  time
0         0.448924    0.028780    0.005415    00:28
epoch     train_loss  valid_loss  error_rate  time
Updating lock file 'dvc.lock'

Running stage 'test':
> python src/test.py
Updating lock file 'dvc.lock'

To track the changes with git, run:

    git add dvc.lock
```

Set up remote storage (S3)

Create an AWS Bucket. Ensure you have access to it (`aws s3 ls`)

```
$ dvc remote add myremote s3://mybucket
$ dvc remote default myremote
```

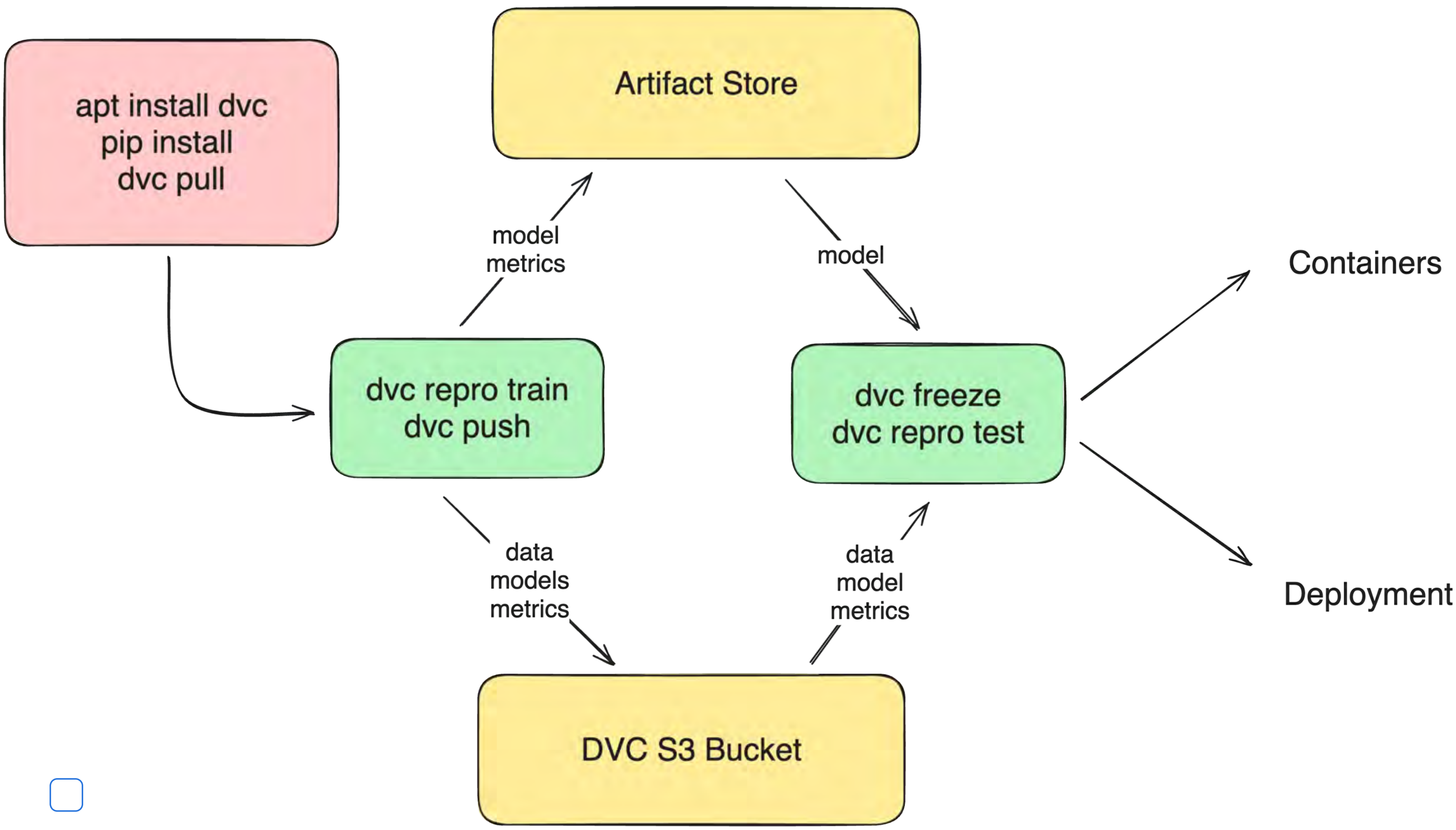
Workflow using git, DVC and DVC remotes



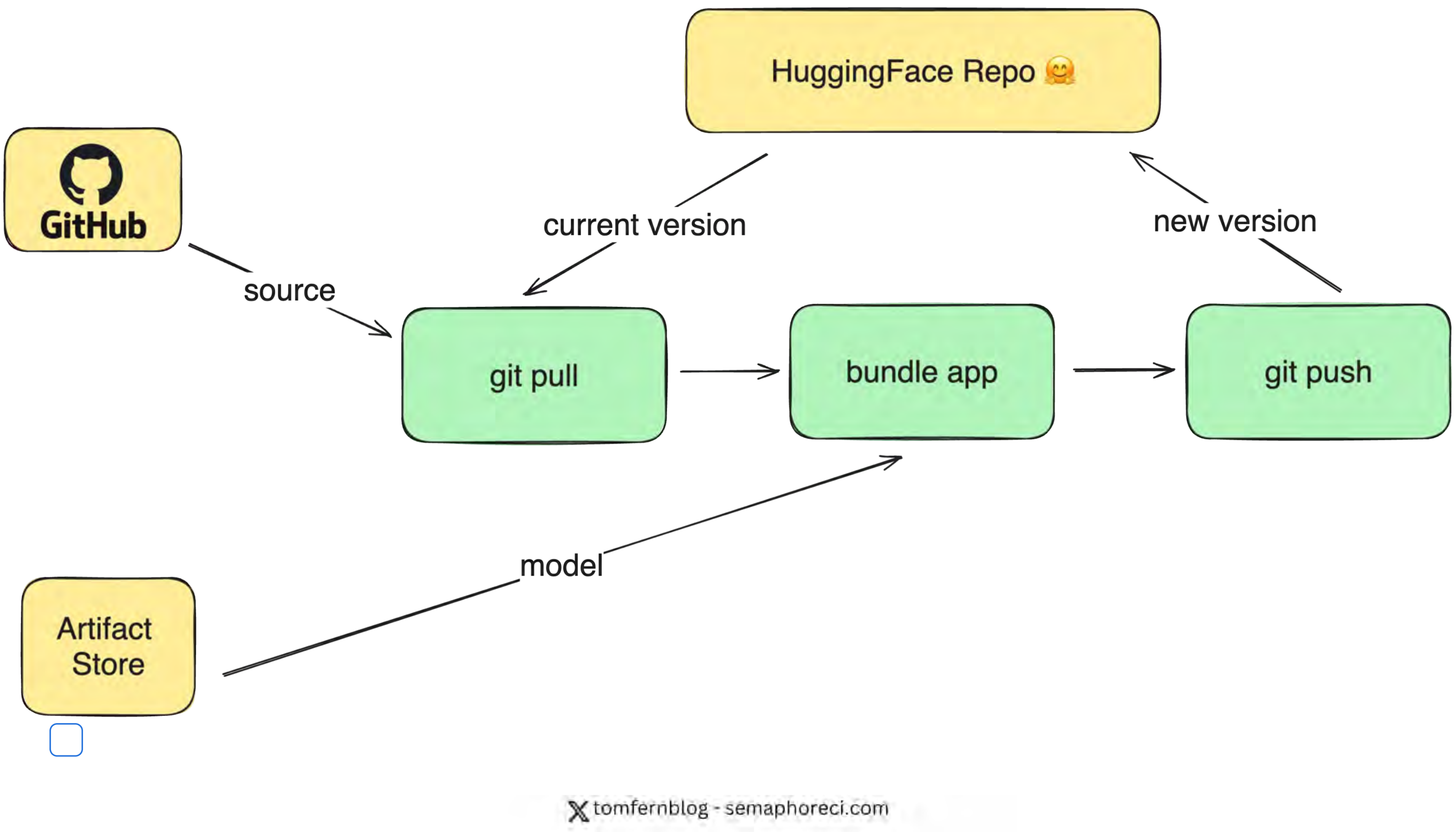
Using the remote

```
# get latest data
$ git pull origin main
$ dvc pull
# run experiments
$ dvc repro
# push changes
$ git add dvc.lock dvc.yaml
$ git commit -m "run experiment X"
$ git push origin main
$ dvc push
```

Continuous Integration
Workflow for DVC and S3



Continuous Delivery Workflow using HuggingFace



Deploy script for HuggingFace

You need:

- A HugginFace account
- Upload the public SSH key to HuggingFace
- Create an space on HuggingFace

Steps:

1. Clone HuggingFace repository
2. Clone Git Repository
3. Pull DVC Data
4. Copy code and models into HuggingFace repository
5. Push changes into HuggingFace
6. Deployment takes place automatically


Thank You!
Tommy Fernandez
Technical Writer at Semaphore
contact@tomfern.com
/TomFernBlog