Lecture 21: Operations - Automation - Review

AC215

Pavlos Protopapas
SEAS/ Harvard

Outline

1. Motivation
2. Automation

Outline

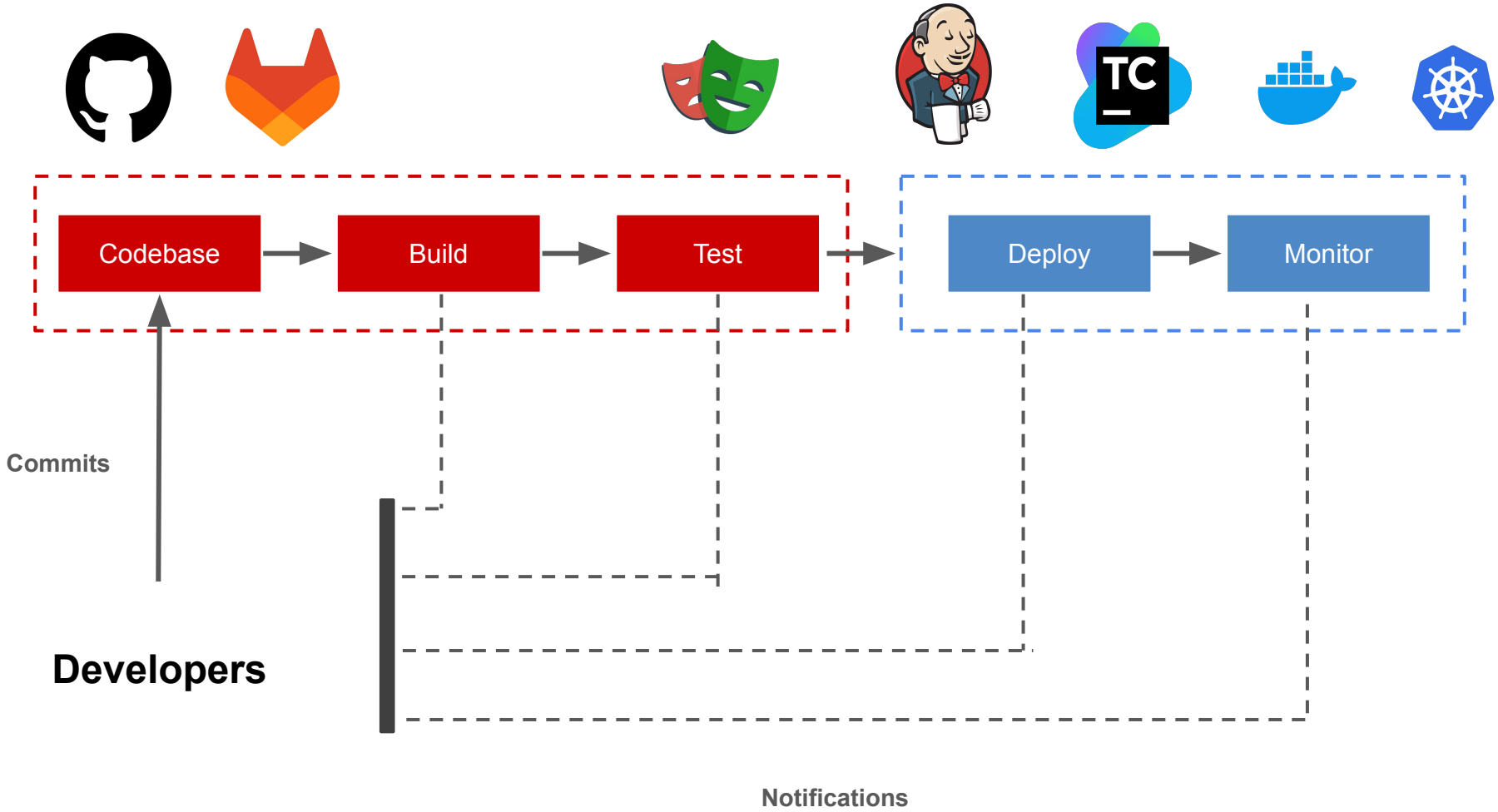
1. **Motivation**
2. Automation

Automation

We want to automate the following operations:

- Running data pipeline jobs
- Model training & model deployment
- Frontend deployment
- Backend deployment
- Building & Pushing Docker containers

CI / CD


Codebase
Build
Test
Deploy
Monitor
Commits
Developers
Notifications

Continuous Integration (CI)

Continuous Integration (CI) **automates** a series of scripts to run **whenever** changes are pushed, in order to:

- Continuously integrate changes into the production branch
- Run automated tests
- Enforce coding standards
- Perform static code analysis

Note: Coding standards are about style and consistency.

Static code analysis is about identifying potential defects and vulnerabilities.

Continuous Integration (CI)

Continuous Integration (CI) allows multiple developers to contribute to a shared repository by **automating** quality checks to ensure the code remains functional.

This is achieved through:

- Running automated tests to catch bugs early
- Enforcing coding standards to maintain consistency
- Performing static code analysis

Note: Coding standards are about style and consistency.

Static code analysis is about identifying potential defects and vulnerabilities without executing the code.

Continuous Deployment / Delivery (CD)

Continuous Deployment (CD): Is to take automation further by **deploying** code changes to production automatically, as soon as the new features are **integrated** into the main codebase.

Continuous Delivery (CD): extension of CI to ensure software can be reliably released at any time.

Note: Delivery ensures changes are always ready to be deployed, but keeping a person in the loop. Deployment ensures changes are automatically passed into production.

CI in AI/ML

Regularly integrates **ML code**, **data**, and **models** into a shared repository while ensuring consistency through automated checks.

Automatically tests these integrations to ensure consistency.

Enables **quick detection of issues** in:

- Data quality
- Code correctness
- Model performance

Challenges in AI/ML CI

Balancing **frequent integrations** with:

- **High computational demands** of training models.
- **Long-running tests** for large datasets and complex pipelines.

Ensuring compatibility between evolving **data**, **features**, and **model versions**.

CI in AI/ML

Regularly integrates **ML code**, **data**, and **models** into a shared repository while ensuring consistency through automated checks.

Enables **quick detection of issues** related to:

- Data quality
- Code correctness
- Model performance

Challenges in AI/ML CI

Balancing **frequent integrations** with:

- **High computational demands** of training models.
- **Long-running tests** for large datasets and complex pipelines.

Ensuring compatibility between evolving **data**, **features**, and **model versions**.

CD in AI/ML

Automates the release of **ML models, data, and pipelines** to production environments, ensuring that updates are deployed as soon as possible, maintaining reliability after passing all validation steps.

Benefits:

- All the CI benefits
- Performance validation: Ensures metrics are within tolerance
- Scalability: Leverages cloud infrastructure

Challenges:

- Data drift
- Validation of complex models such as LLMs
- Cost

CI/CD Providers and Tools

Some of the common CI CD provider and tools:

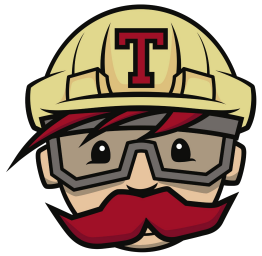
TeamCity

Github

GitLab

Jenkins

CircleCI



TravisCI

How to Implement CI/CD

We already have a deployment container that can:

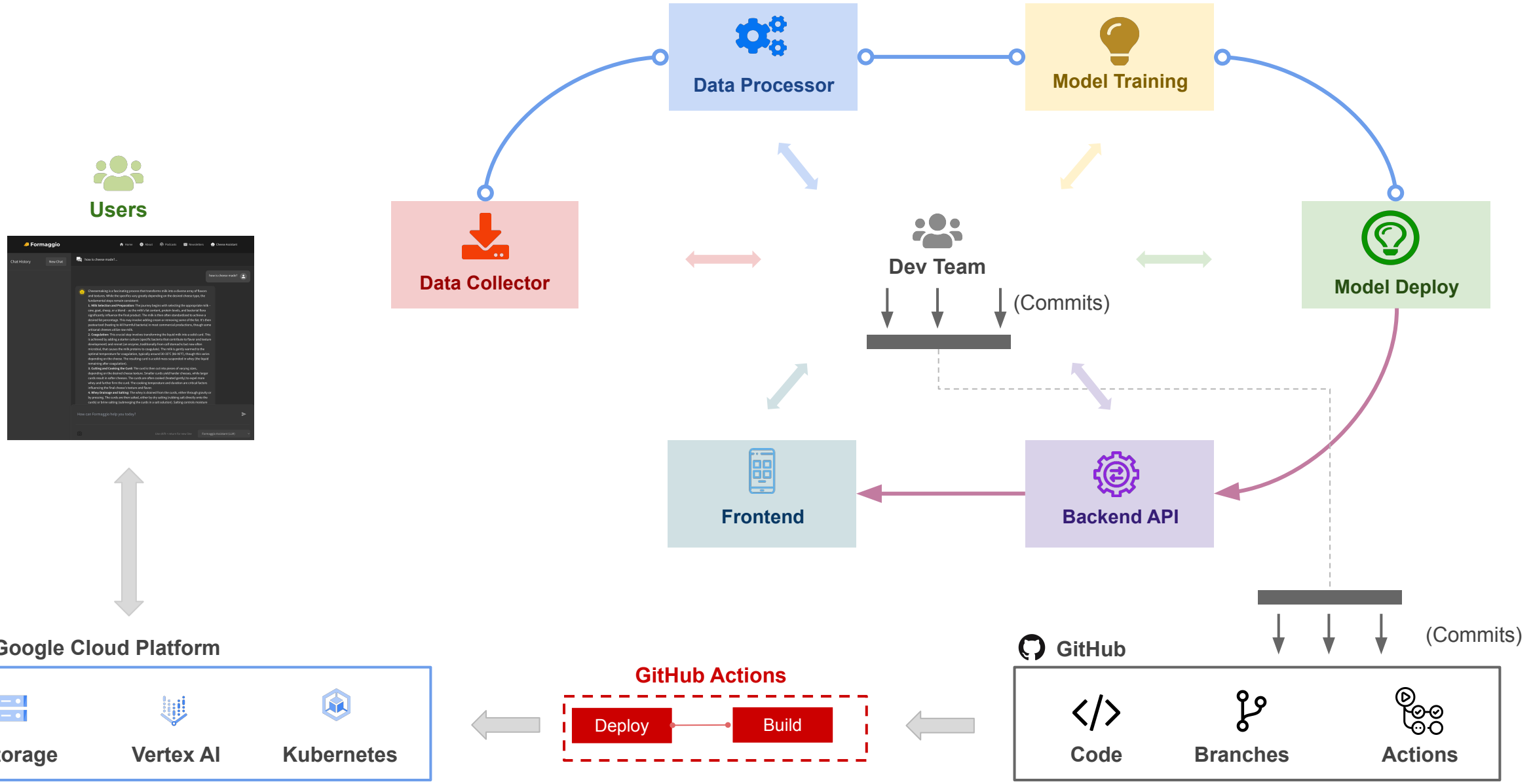
- Build Docker images.
- Run Vertex AI pipeline jobs.
- Deploy app to K8s cluster.

How to Implement CD

We can automate using GitHub Actions by:

- Monitoring code commits.
- Build & run **deployment container**.
- Invoke CLI in deployment container to:
  - Build & push docker images for release
  - Run Vertex AI jobs using new newly build images
  - Deploy newly build images of app to K8s cluster

Cheese App: CI CD



Tutorial: Continuous Integration, Continuous Deployment

Steps to apply **CI / CD** on the cheese app components:

- Create a Github / Workflow file defining deployment steps

  https://github.com/dlops-io/cheese-app-v4/tree/main/.github/workflows

- Commit your code using /deploy-app to the commit message

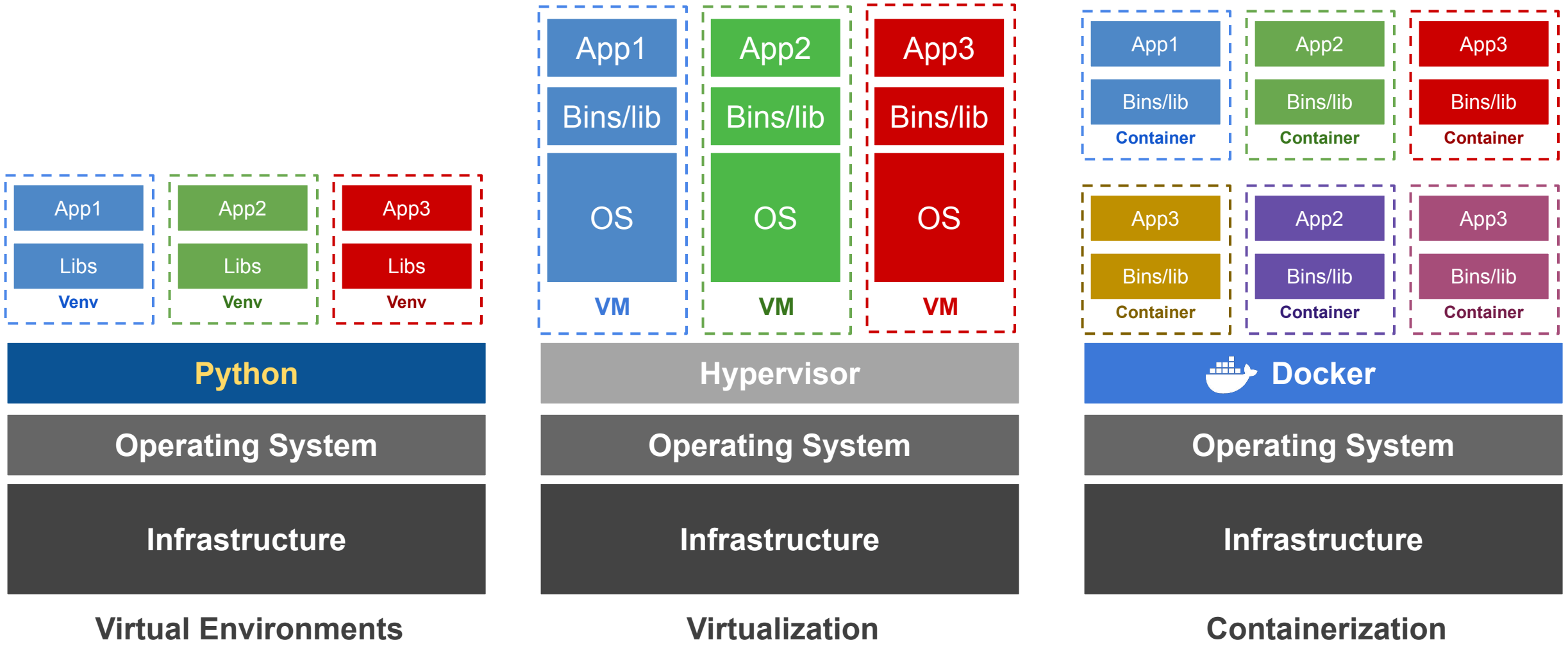
```
# Basic format
git commit -m "your message /deploy-app"
```

For detailed instructions, please refer to the following link

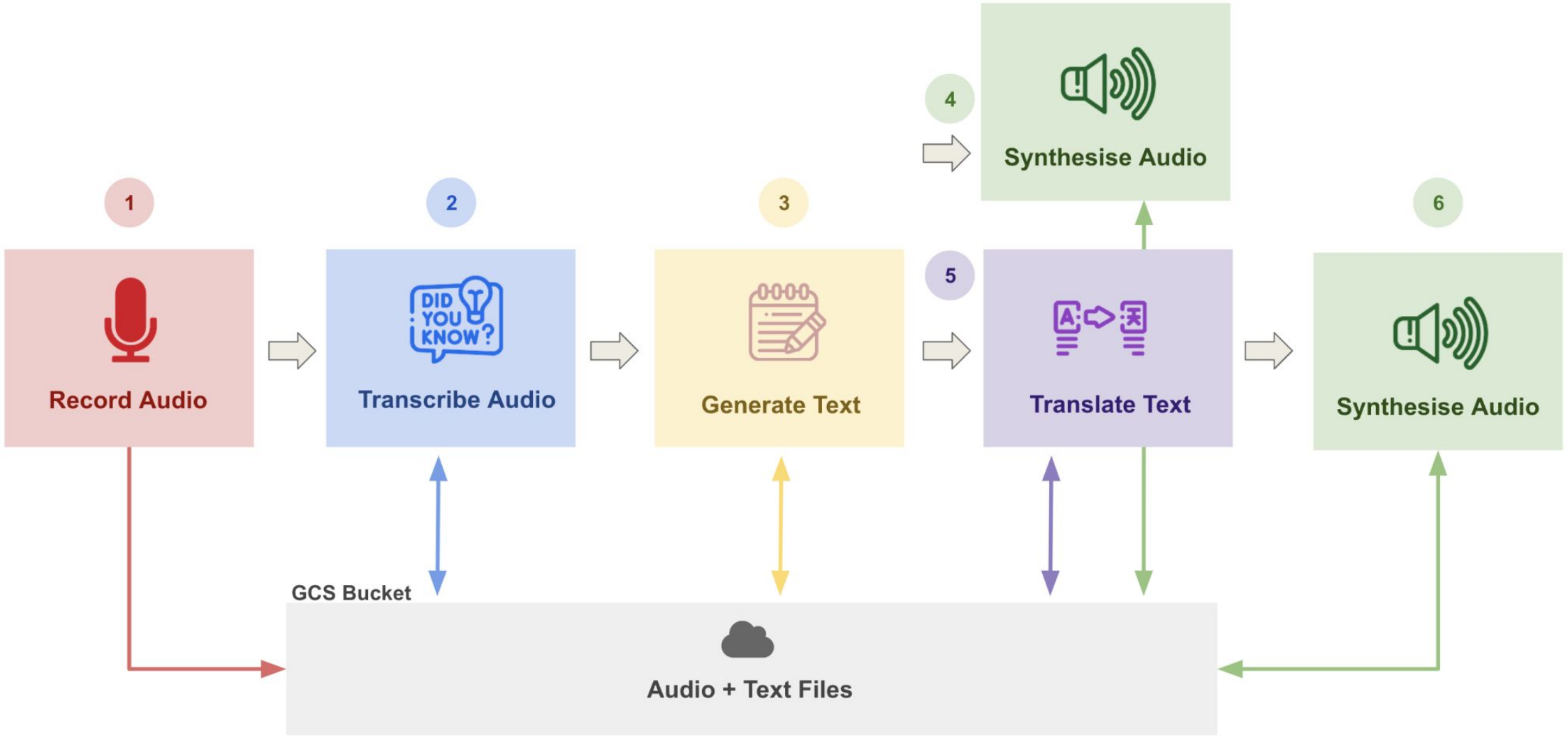
- Cheese App CICD. (https://github.com/dlops-io/cheese-app-v4#cheese-app---automatic
- Cheese App - GitHub Actions. (https://github.com/dlops-io/cheese-app-v4/actions)

Review

Lecture 2-4: Environments vs Virtualization vs Containerization

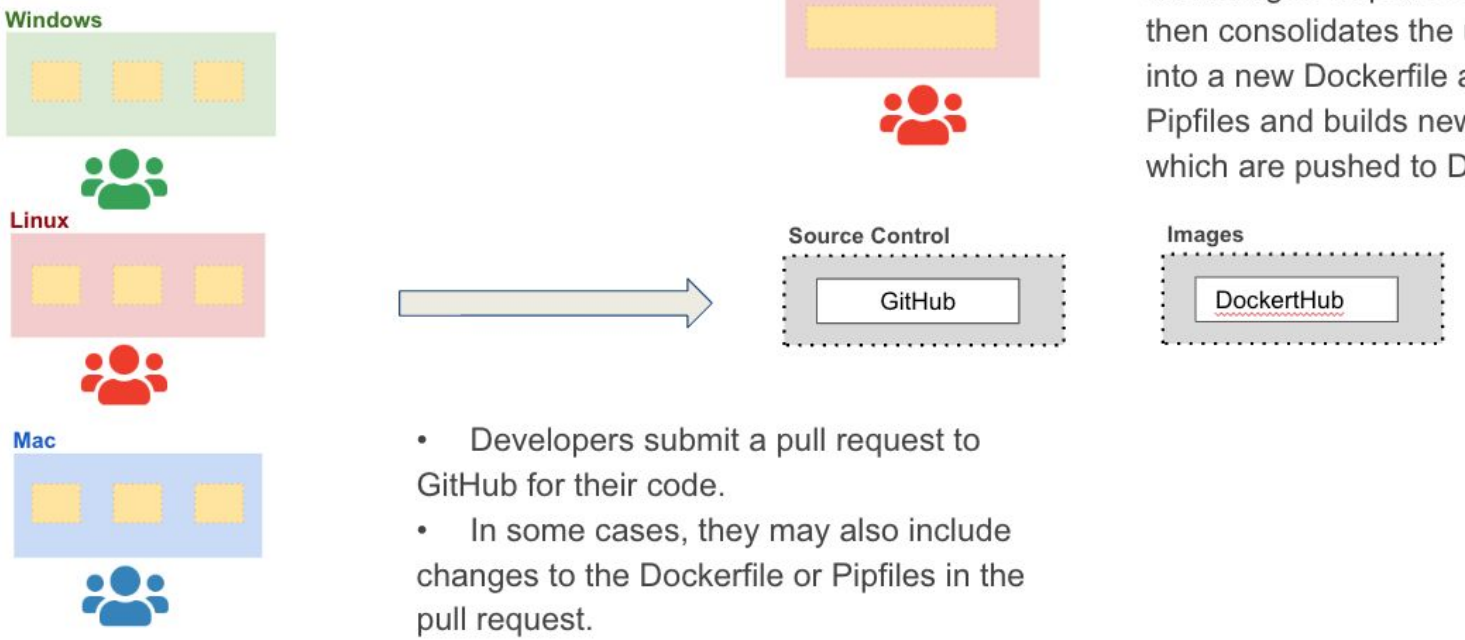


Lecture 2-4: Environments vs Virtualization vs Containerization



Lecture 5: Docker Workflows

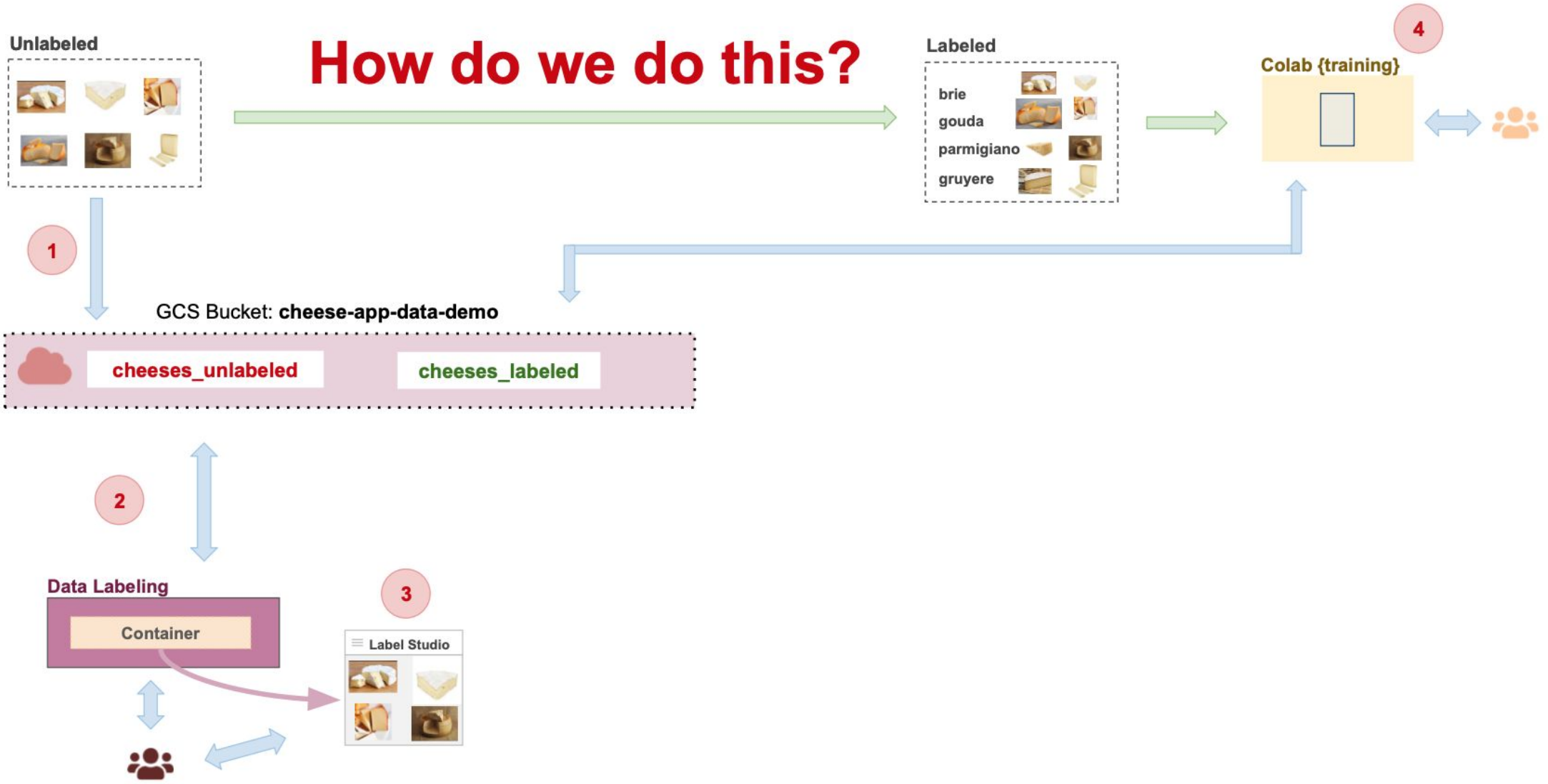
Workflow with Docker: Scenario 2 (later stages of development)



The senior developer reviews and merges all pull requests, then consolidates the updates into a new Dockerfile and Pipfiles and builds new images which are pushed to DockerHub.

- Developers submit a pull request to GitHub for their code.
- In some cases, they may also include changes to the Dockerfile or Pipfiles in the pull request.

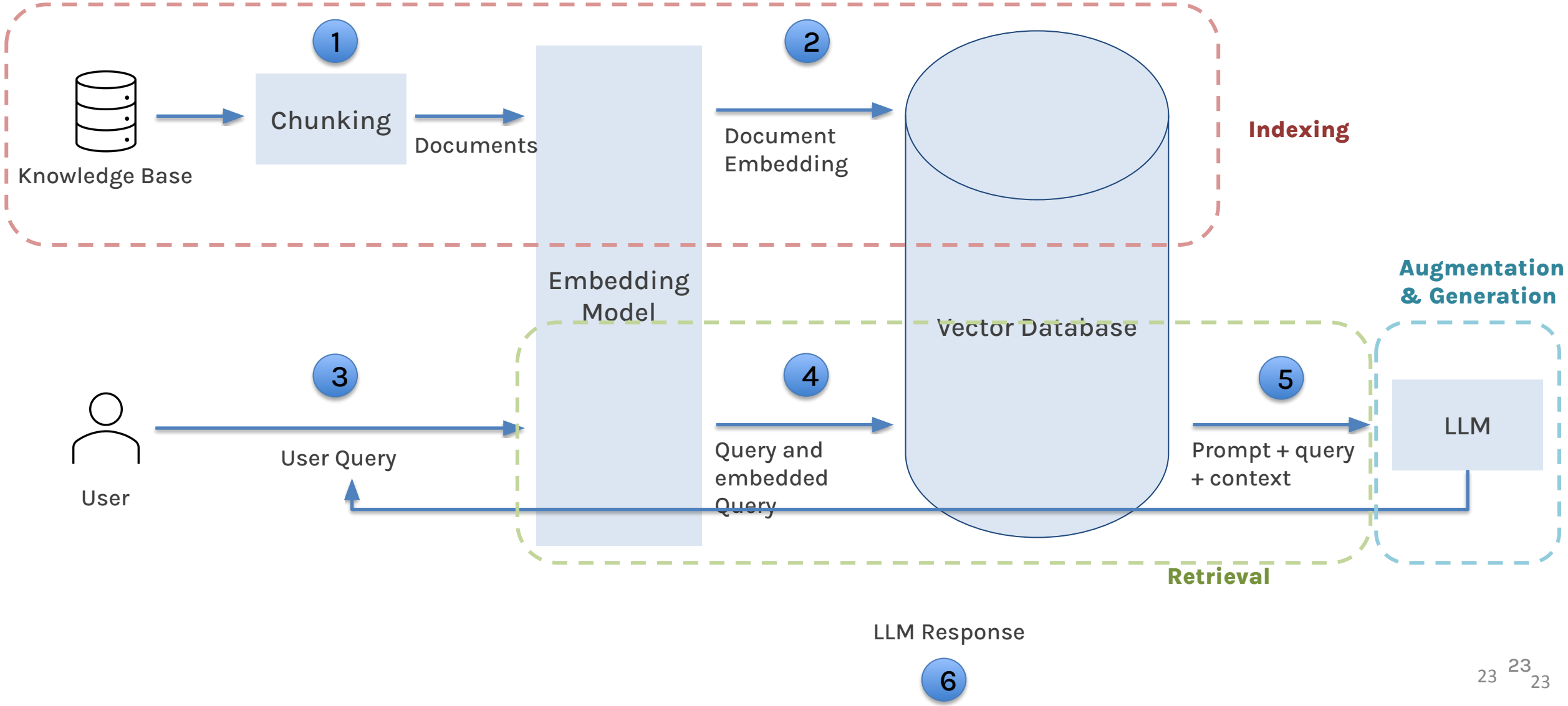
Lecture 6: Data Labeling and Data Versioning



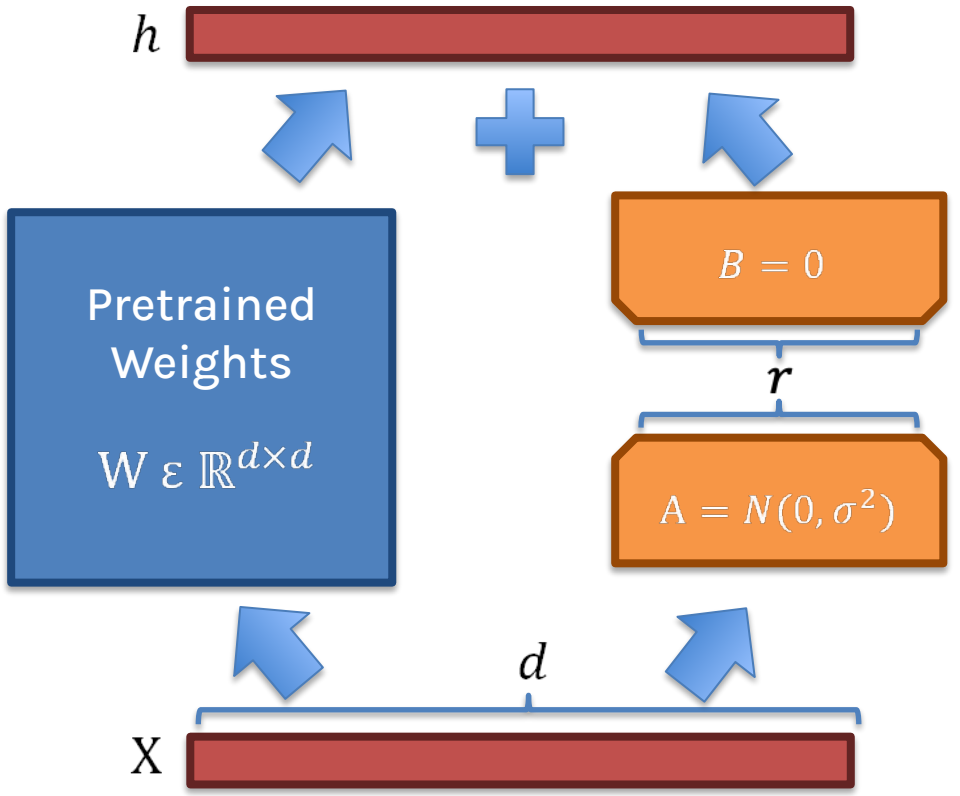
Lecture 7: Instruction based GPT => ChatGPT

We will break it down into 3 steps:

Lecture 8-9: RAGs



Lecture 9-10: Fine Tuning LLMs (LoRA, qLoRA)

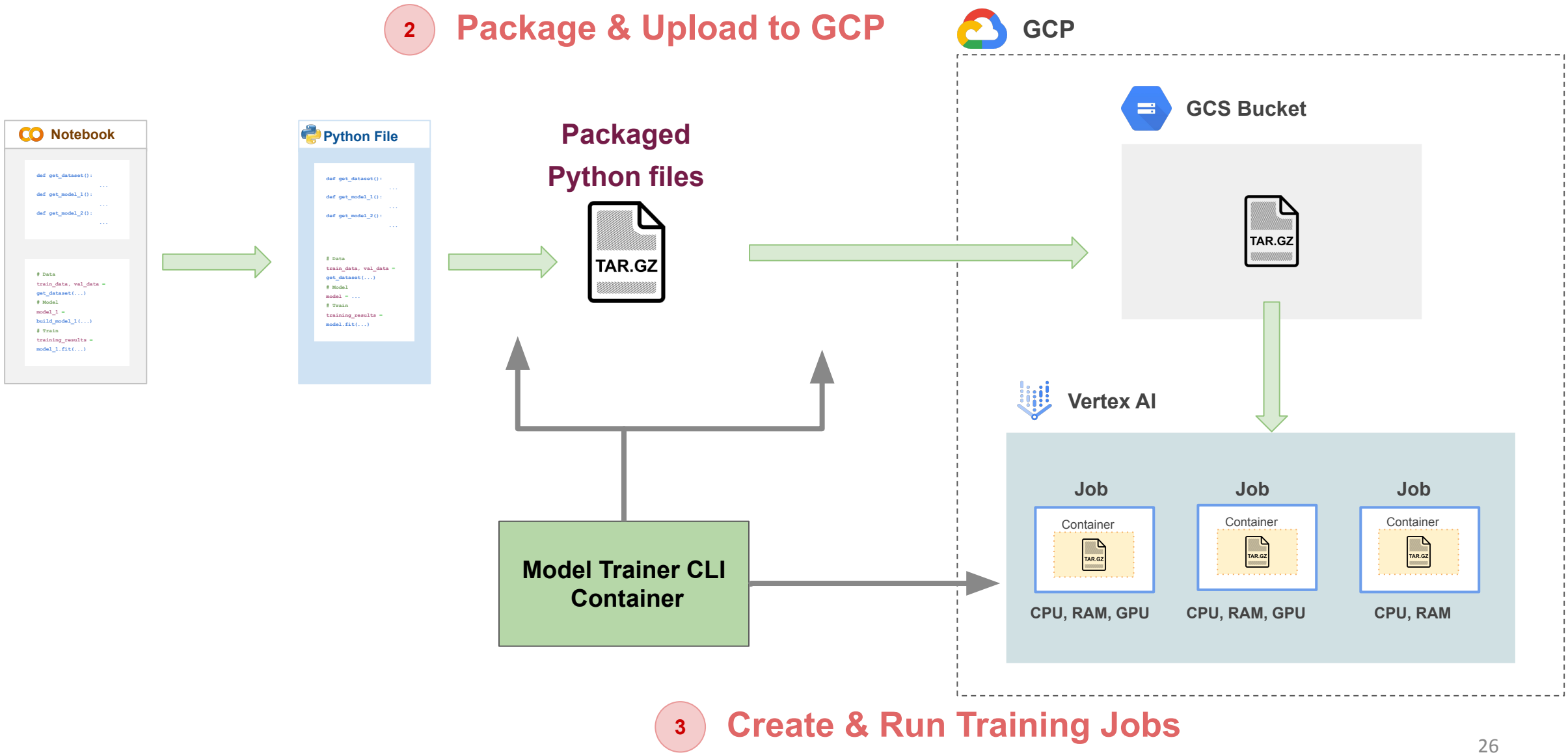


Notice how the reparameterization (LoRA) runs parallel to the original model.

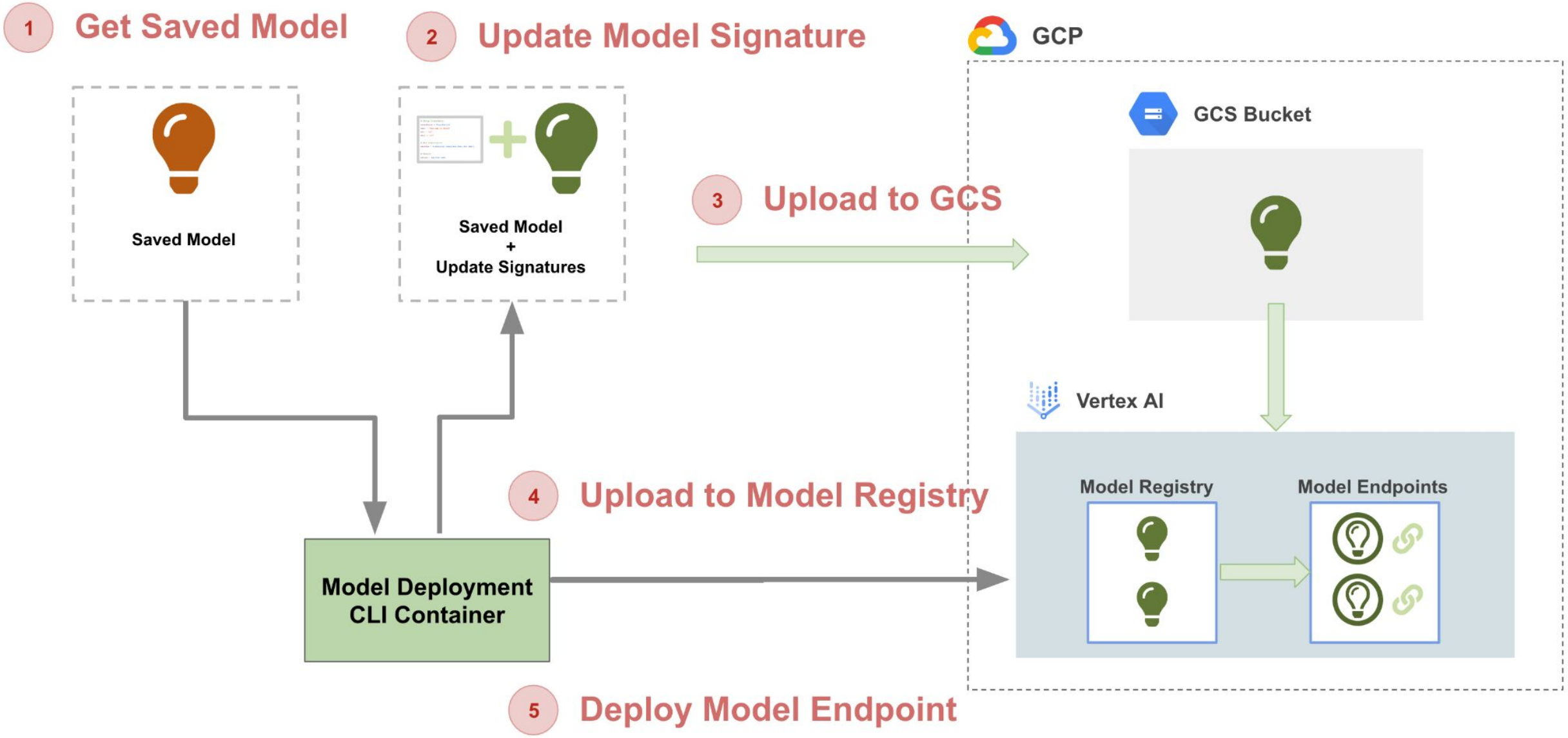
24

Lecture 11: Distillation

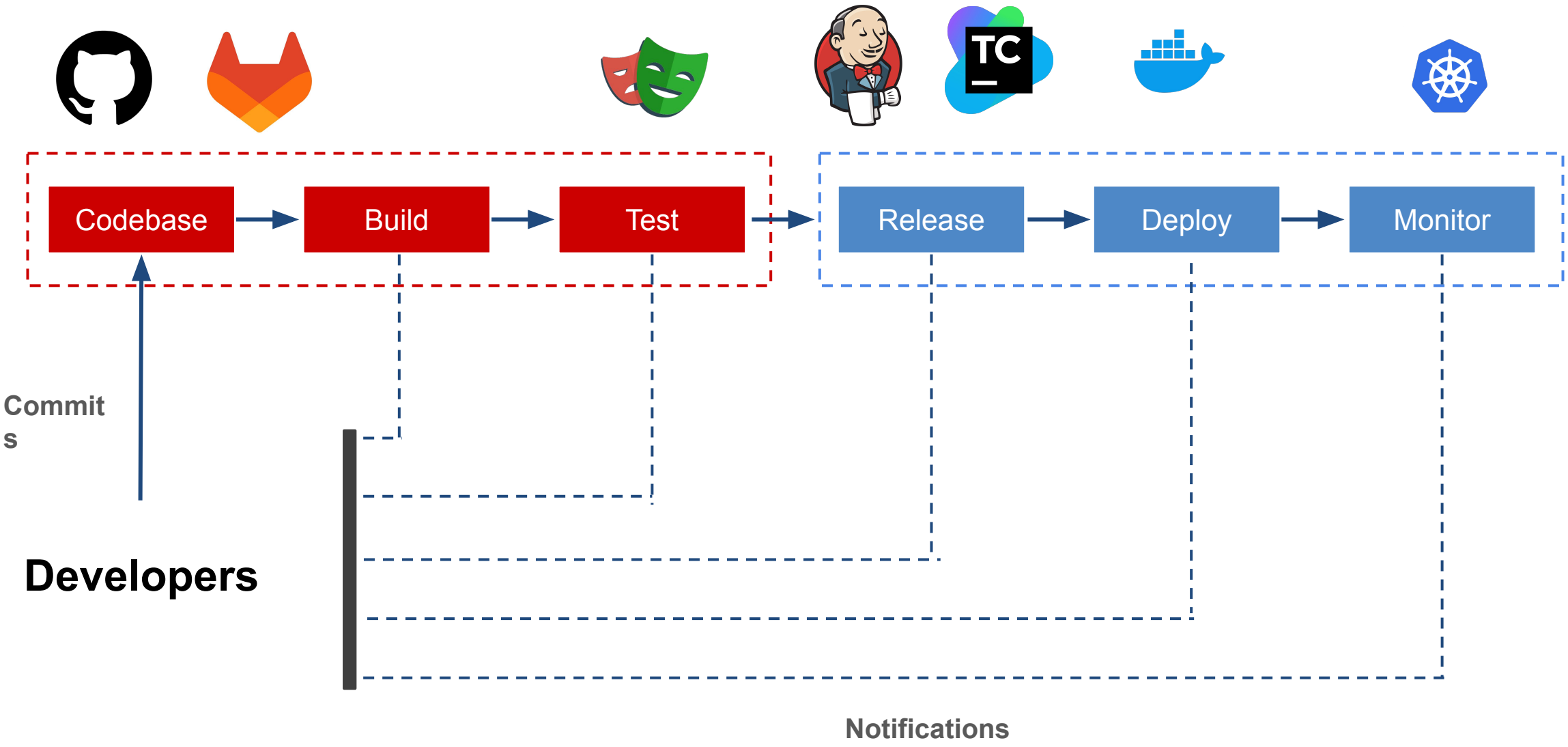
Lecture 12-13: Weight and Biases/VertexAI serveless training



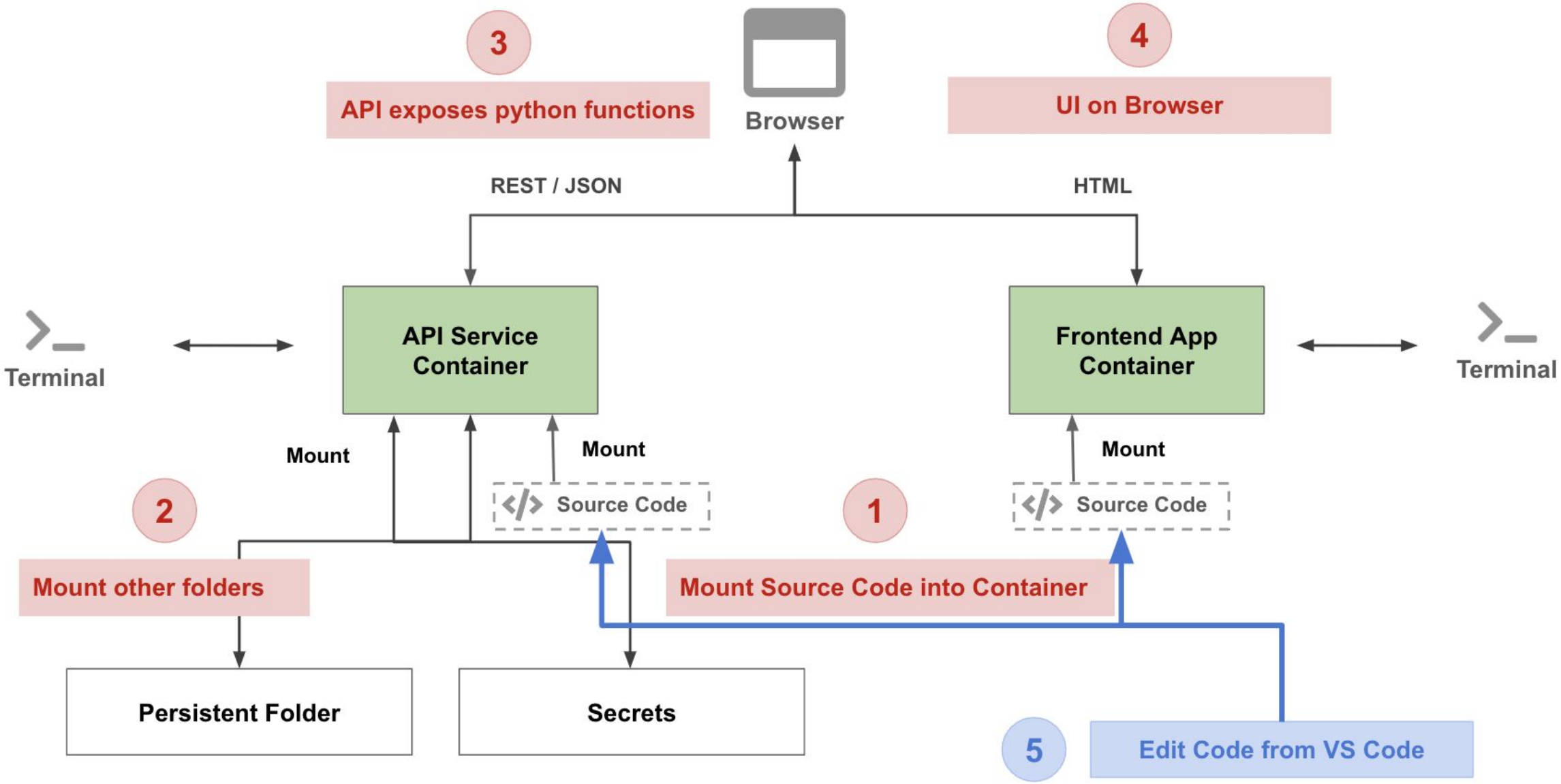
Lecture 15: Model Deployment with VertexAI



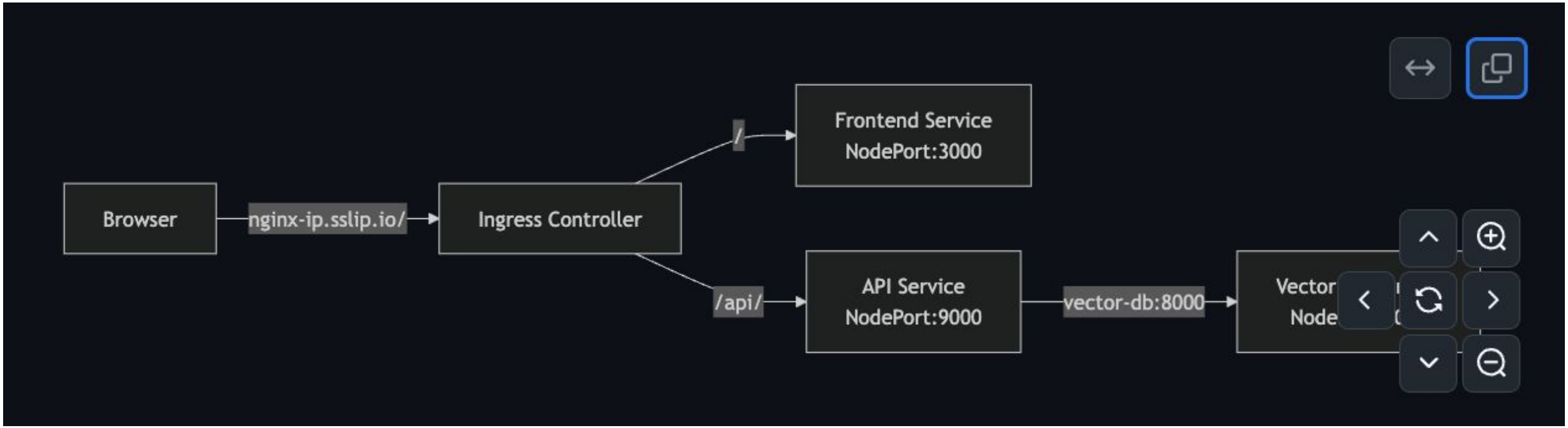
Lecture 16: CI



Lecture 17-18: Frontend, Fast API



Lecture 19: Ansible



THANK YOU