ENTERPRISE SYSTEMS INTEGRATION

ACIT4850 – WINTER 2024

AGENDA – LESSON 10

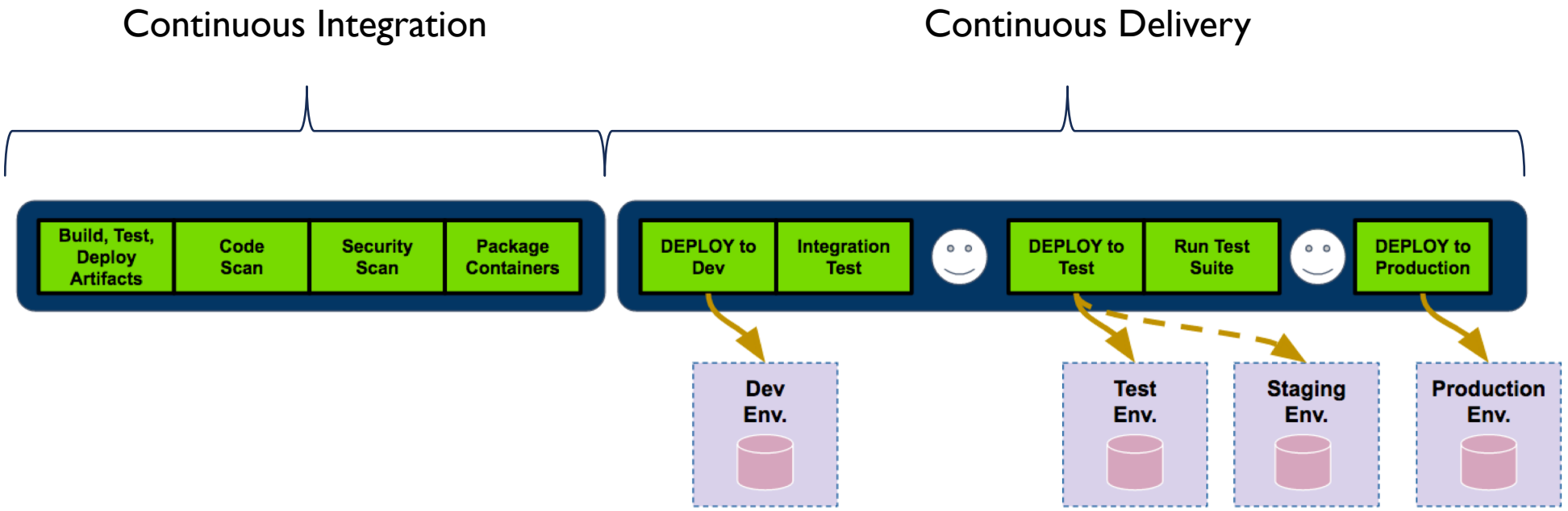
- Quiz 9 on D2L
- Topics
  - Continuous Delivery/Deployment
  - Development Workflows
- Lab 10 – Workflows and Deployment
  - Conditional Stages and Pipeline Parameters

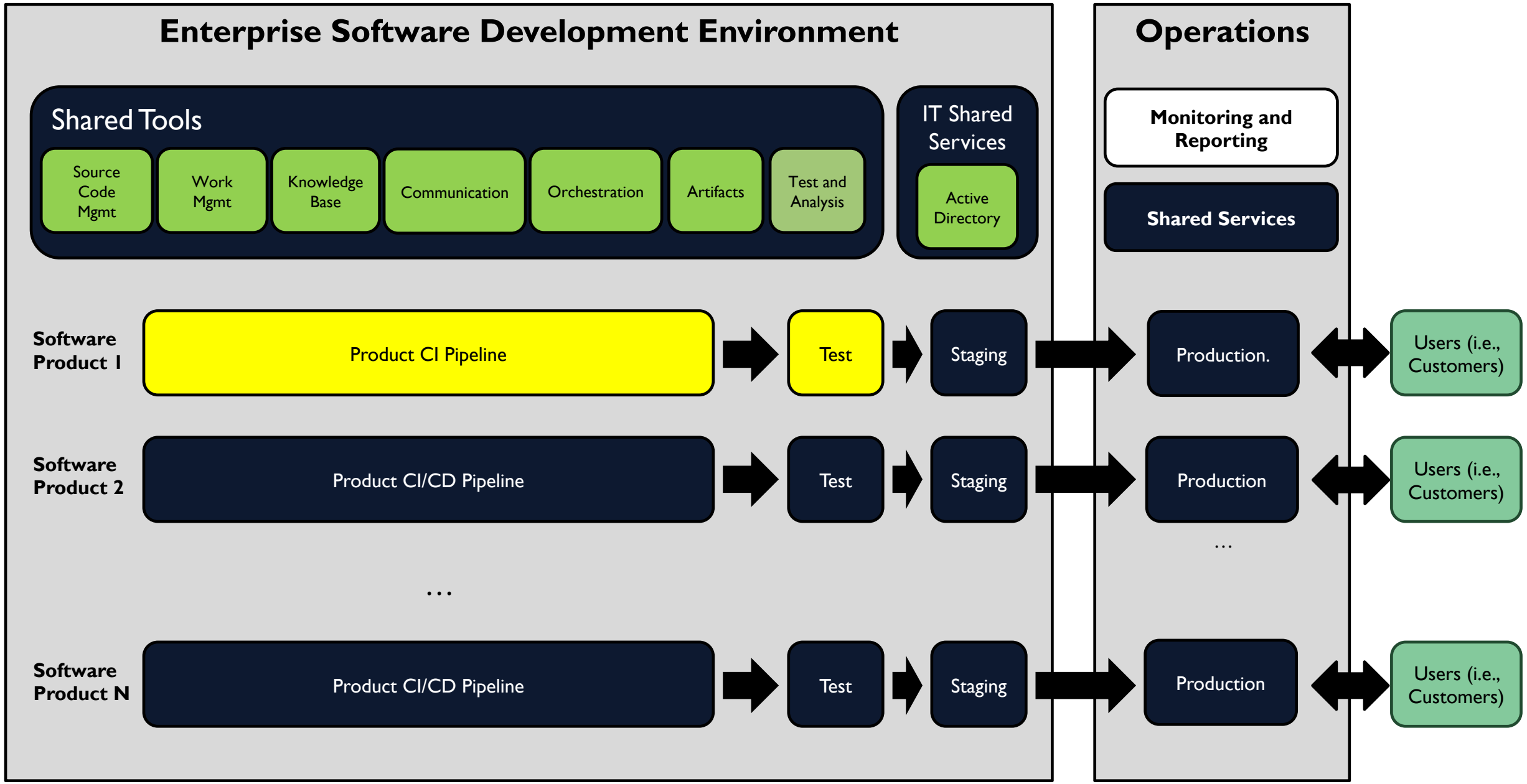
QUIZ 9

- Open Book – You can use the Lesson 10 reading materials
- You have 15 minutes to complete
- We will resume in 15 minutes, or when everyone is done

CI/CD PIPELINE

To enable Continuous Delivery, we need a pipeline. This is effectively the pipeline we have been creating in this course.




Enterprise Software Development Environment
Shared Tools
Source Code Mgmt
Work Mgmt
Knowledge Base
Communication
Orchestration
Artifacts
Test and Analysis
IT Shared Services
Active Directory
Operations
Monitoring and Reporting
Shared Services
Software Product 1
Product CI Pipeline
Test
Staging
Production.
Users (i.e., Customers)
Software Product 2
Product CI/CD Pipeline
Test
Staging
Production
Users (i.e., Customers)
...
...
Software Product N
Product CI/CD Pipeline
Test
Staging
Production
Users (i.e., Customers)

THE ROADMAP (AKA COURSE SCHEDULE)

Week	Topics	Notes
1	• Components of an Enterprise Development Environment • Software Source Code Management	Lab 1
2	• Work Management and Knowledge Base Tools	Lab 2, Quiz 1
3	• Tool Selection – Requirements • Integration and Security	Lab 3, Quiz 2
4	• Tool Selection – Stakeholders/Process • Continuous Integration (CI) Tool • CI Tool Setup	Lab 4, Quiz 3
5	• CI Pipelines – Python	Lab 5, Quiz 4
6	• CI Pipelines – Shared Libraries	Lab 6, Quiz 5, Assignment 1 Due
7	• CI Pipelines – Java and Static Code Analysis *Note: At home lab for Monday set*	Lab 7, Quiz 6 (Sets A and B)
8	• Midterm	Midterm Review Quiz
9	• CI Pipelines – Alternate Tools (GitLab CI)	Lab 8, Quiz 6 (Set C), Quiz 7
10	• Spring Break	
11	• Continuous Delivery (CD) • CD Pipelines - Containerization	Lab 9, Quiz 8, Assignment 2 Due
12	• CD Pipelines – Deployment • Developer Workflows	Lab 10, Quiz 9
13	• CD Pipelines – Alternate Tools (GitLab CI) • *Note: At home lab for Monday Set*	Lab 11, Quiz 10 (Sets A and B)
14	• Microservices Pipelines • Final Exam Preview	Quiz 10 (Set C), Assignment 3 Due
15	**Final Exam**	

CONTINUOUS DELIVERY VS CONTINUOUS DEPLOYMENT

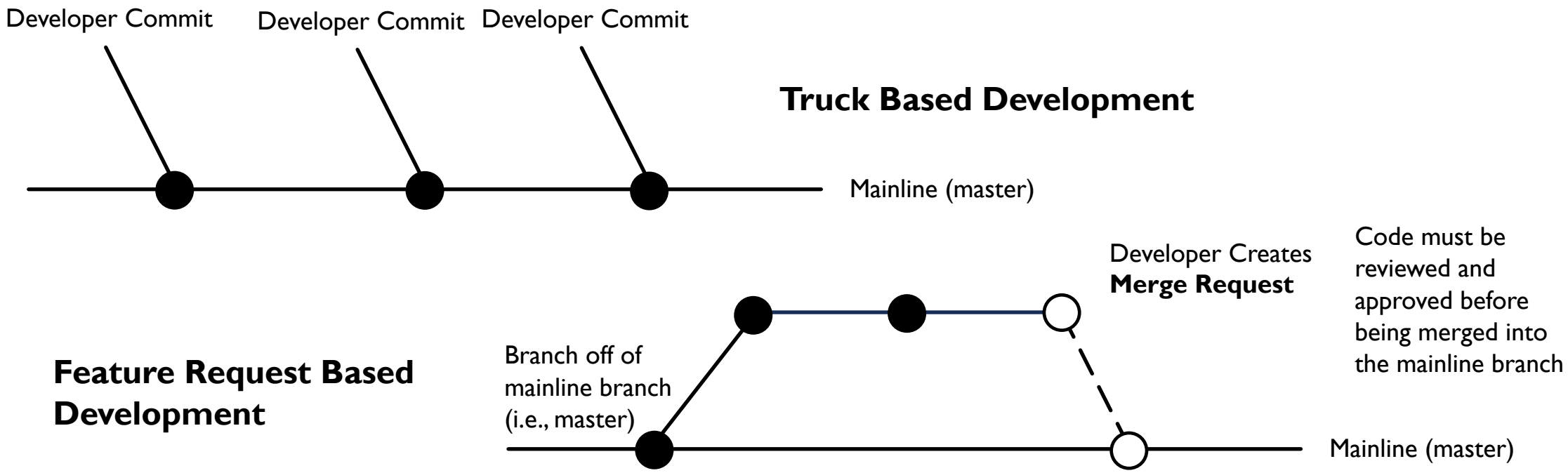
- **Continuous Delivery** – You build, test and package the software into a **deployable state** each time the pipeline is run (ideally triggered by a developer commit). And you may deploy it to internal target environment for integration and testing (i.e., development and test environments). *Deployment to a Production environment is still manual or at best manually triggered.*
  - For our last lab, the deployable state was a Docker image

- **Continuous Deployment** – You build, test, package, stage and deploy the software into Production automatically each time a change is committed.

Continuous Deployment is generally very risky to do unless you invest in sufficient automated testing and deployment infrastructure.
It is typically reserved for large consumer oriented SaaS products – like Facebook, Google, etc.

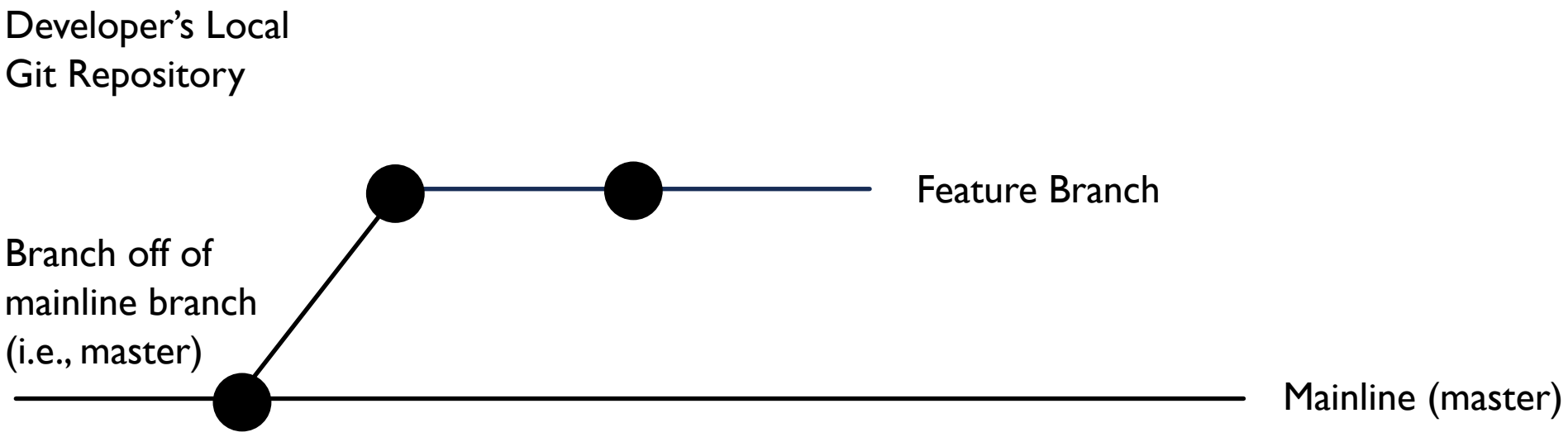
DEVELOPMENT WORKFLOWS

- Trunk based development – Developers commit changes directly on Master
- Feature Branching – Developers commit changes to a Feature Branch and use a Merge Request as a checkpoint (code review, build) before merging into Master

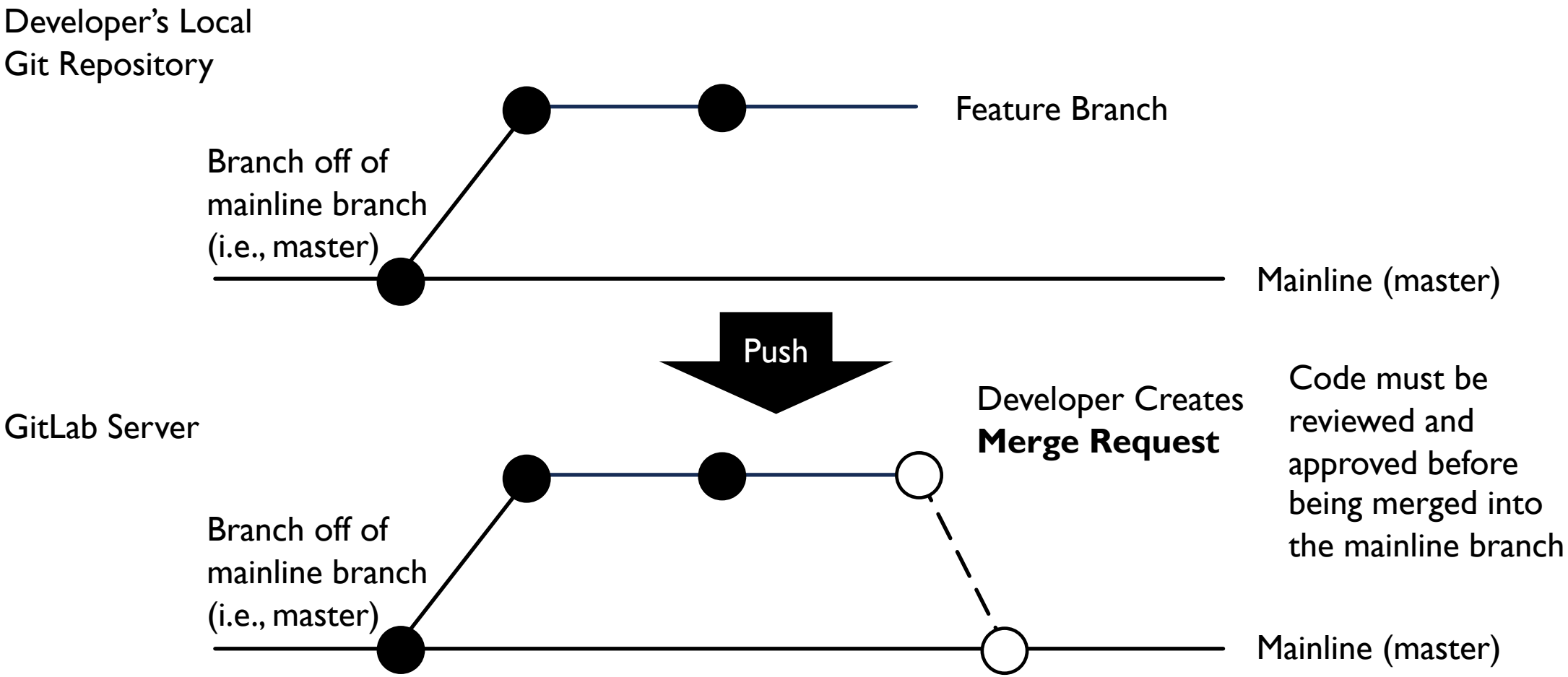


DEVELOPMENT WORKFLOWS – FEATURE BRANCHES

- Create a local branch in your Git repository to work on a specific feature (or bug).

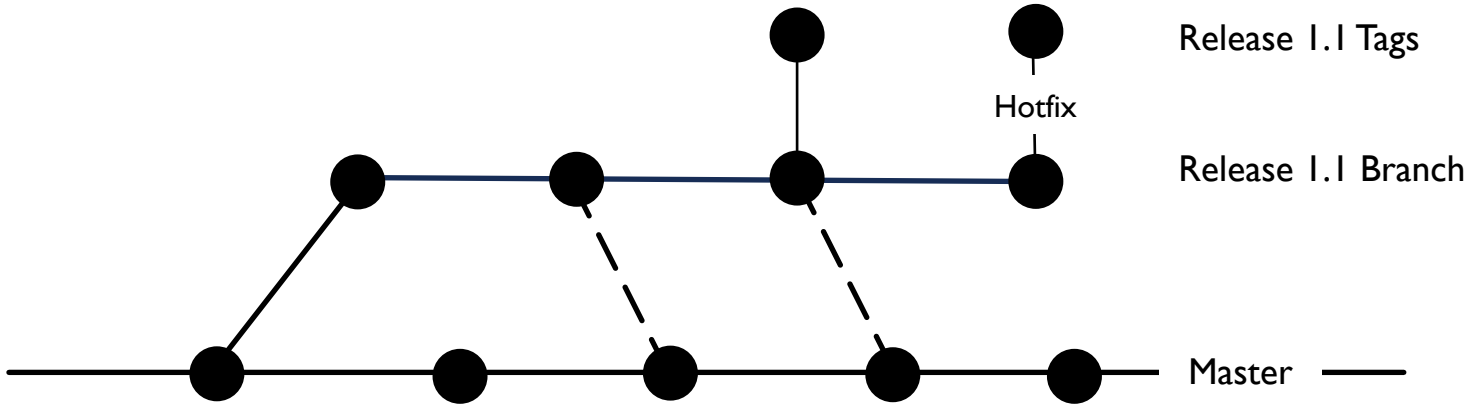


DEVELOPMENT WORFLOWS – MERGE REQUEST



DEVELOPMENT WORKFLOWS

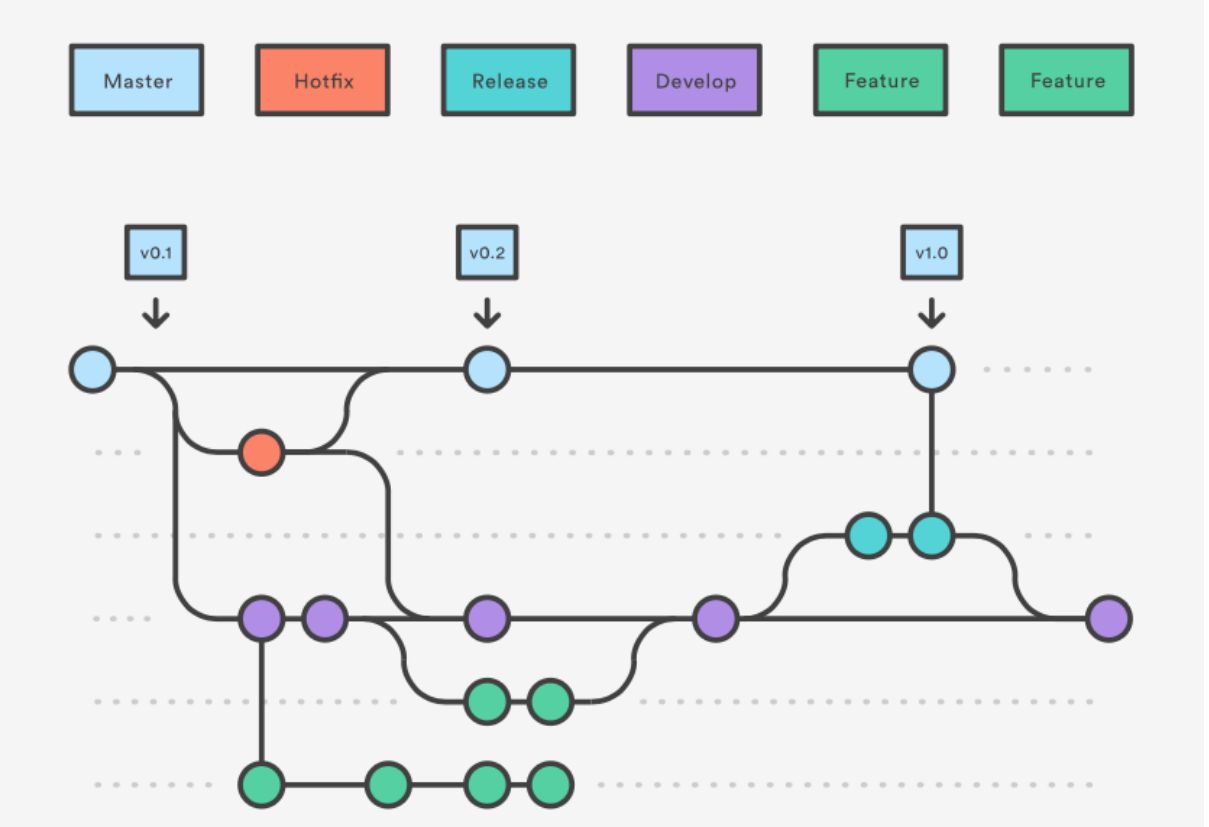
- **Release Branches** – Where work for a specific release is done
  - Typically when you are getting ready for a release, you create a branch to do work for that release
  - Then work on subsequent releases can occur in parameter on other branches (i.e., master)
- **Tags** – Snapshot of the source code for a specific release. This is what released out to product and never is changed.
- **Hotfixes** – Quick fixes for issues in production. Typically are on the Release Branch and then tagged.



DEVELOPMENT WORKFLOW - OTHERS

GitFlow – Work is done in Feature Branches off of a Development branch. Releases are merged into Master.

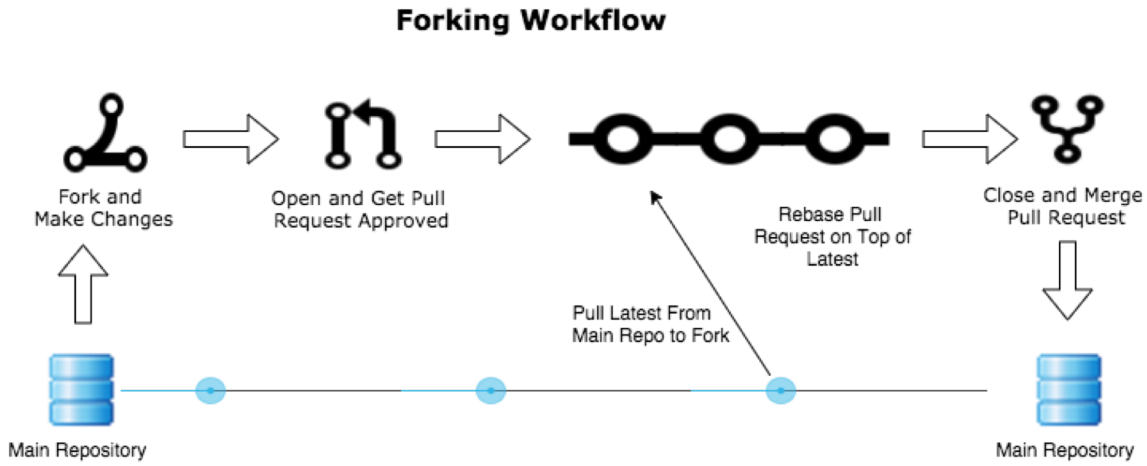
- Master is the source of truth for the current release in production
- Good for SaaS software products with a scheduled release cycle
- Not good for products with multiple concurrent releases to different customers



DEVELOPMENT WORKFLOW - OTHERS

Forking – Used in open-source projects.

- Develop fork the repository to make their changes
- The fork is their server-side copy (not just local)
- Develop makes changes to their fork (using feature branching)
- Their feature branch is pushed to their server-side copy
- A pull (or merge) request is opened to get approval to merge the change into the original server-side repository



The main advantage of the Forking Workflow is that contributions can be integrated without the need for everybody to push to a single central repository. Developers push to their own server-side repositories, and only the project maintainer can push to the official repository. **This allows the maintainer to accept commits from any developer without giving them write access to the official codebase**.

NEW LAB REQUIREMENTS

- **(Existing) REQ1200** – The Enterprise Development Environment shall use containerization to package applications for deployment to a target environment.
- **(Existing) REQ1230** – The Enterprise Development Environment shall hide and encrypt any passwords or secrets used in the pipeline builds.
- **(New) REQ1210** – The Enterprise Development Environment shall require code reviews of all new features prior to integration into the master (i.e., mainline branch).
- **(New) REQ1220** – The Enterprise Development Environment shall have build pipelines that allow container artifacts to optionally be deployed to enable testing by the test team.

EXISTING LAB REQUIREMENT

- **(Existing) REQ1200** – The Enterprise Development Environment shall use containerization to package applications for deployment to a target environment.

(Done) We will add a dockerfile for each repository, describing the runtime environment for the application.

(Done) We will build a Docker image as part of the CI pipeline, and push it to our DockerHub repository.

(TODO) Run your Docker images as containers in a Deploy stage in the CD portion of the pipeline.

EXISTING LAB REQUIREMENT

- **(Existing) REQ1230** – The Enterprise Development Environment shall hide and encrypt any passwords or secrets used in the pipeline builds.

(DONE) We will use Jenkins credentials to store the credentials in an encrypted manner and to be able to use the credential without displaying it in the logs.

```
node {
  withCredentials([string(credentialsId: 'mytoken', variable: 'TOKEN')]) {
    sh '''
      set +x
      curl -H "Token: $TOKEN" https://some.api/
    '''
  }
}
```

NEW LAB REQUIREMENT

- **(New) REQ1210** – The Enterprise Development Environment shall require code reviews of all new features prior to integration into the master (i.e., mainline branch).

All developers must work in local feature branches for both new features and bug fixing.

We will use the Merge Request feature in GitLab to enable peer review of the changes prior to merging into the master branch.

NEW LAB REQUIREMENT

- **(New) REQ1210** – The Enterprise Development Environment shall require code reviews of all new features prior to integration into the master (i.e., mainline branch).

We will use the "when" block in a Jenkins pipeline to have conditional execution of a pipeline stage. Example:

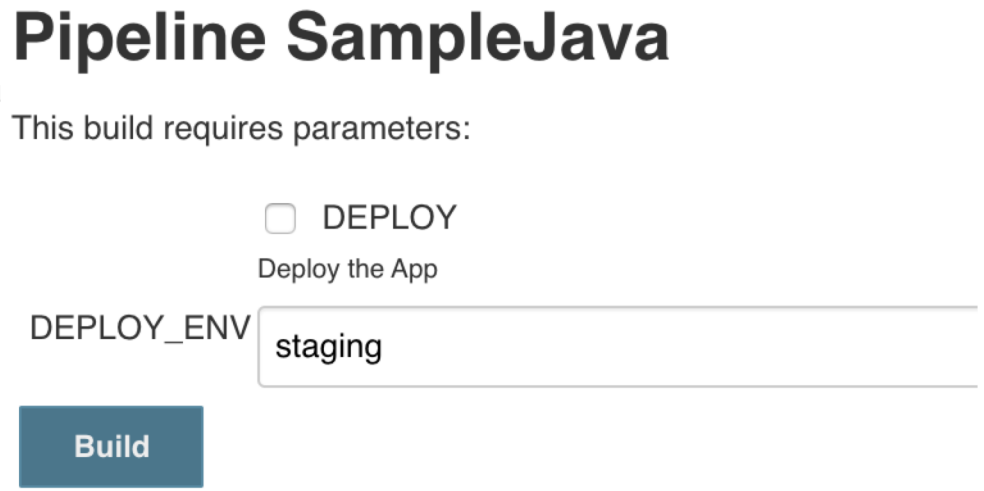
```
// Stage is executed only if the logical expression evaluates to true
stage("Stage Name") {
    when {
        expression { logical expression }
     }
     steps {
        …
     }
}
```

NEW LAB REQUIREMENT

- **(New) REQ1220** – The Enterprise Development Environment shall have build pipelines that allow container artifacts to optionally be deployed to enable testing by the test team.

We will use parameters for this.

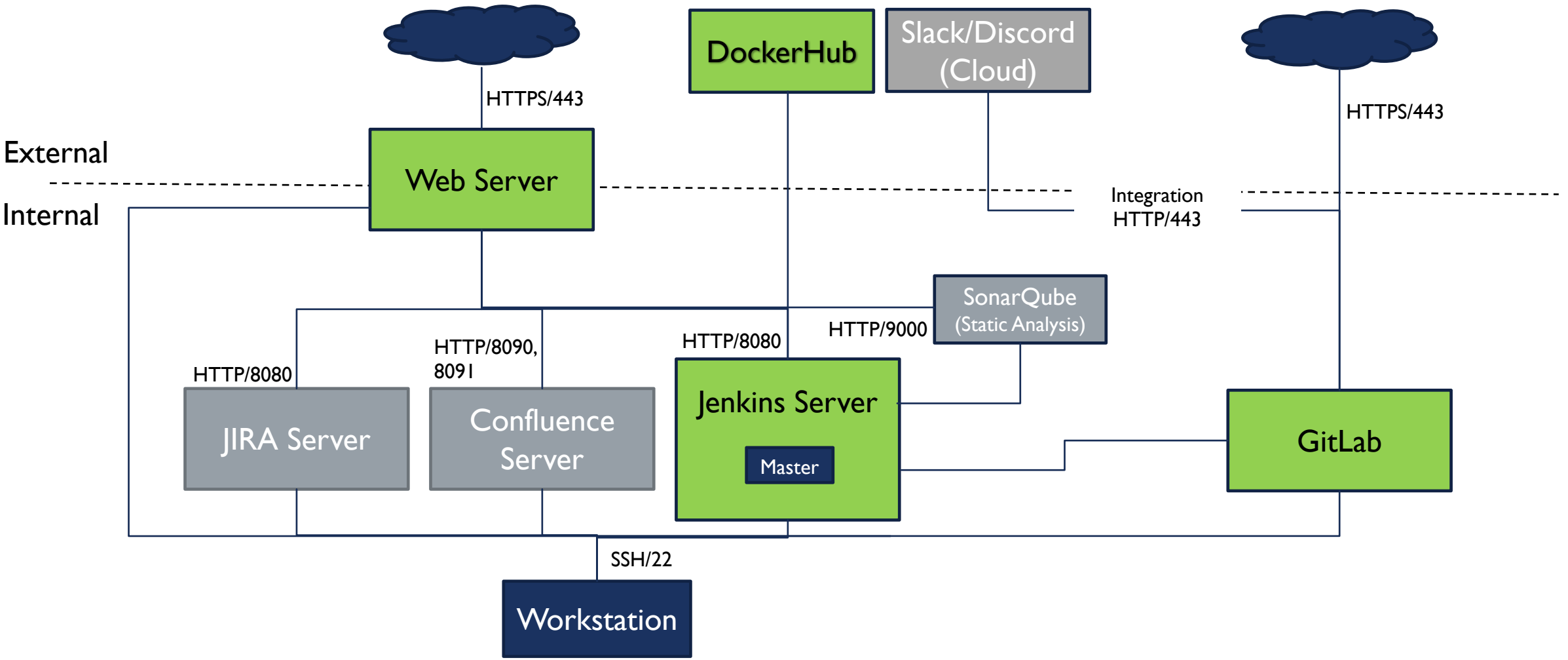
```
pipeline {
    agent any

    parameters {
        booleanParam(defaultValue: false, description: 'Deploy the App', name: 'DEPLOY')
        string(defaultValue: 'staging', description: '', name: 'DEPLOY_ENV')
    }

    stages {
        stage('Build') {
```



References:

- https://jenkins.io/doc/book/pipeline/syntax/#parameters
- https://github.com/jenkinsci/pipeline-model-definition-plugin/wiki/Parametrized-pipelines

YOUR ENTERPRISE DEVELOPMENT ENVIRONMENT (COMPLETED)



TODAY'S LAB

Lab 10

- Posted to D2L
- Work with your partner
- Screenshots and demo due by end of next class