ENTERPRISE SYSTEMS INTEGRATION

ACIT4850 – WINTER 2024

AGENDA – LESSON 4

- Quick Review
- Quiz 3 on D2L
- Topics
  - Tools Evaluations
  - DevOps and Continuous Integration
  - CI Tool – Jenkins
- Lab Requirements
- Lab
  - Demo of Lab 3
  - Start on Lab 4

WARM UP QUESTIONS

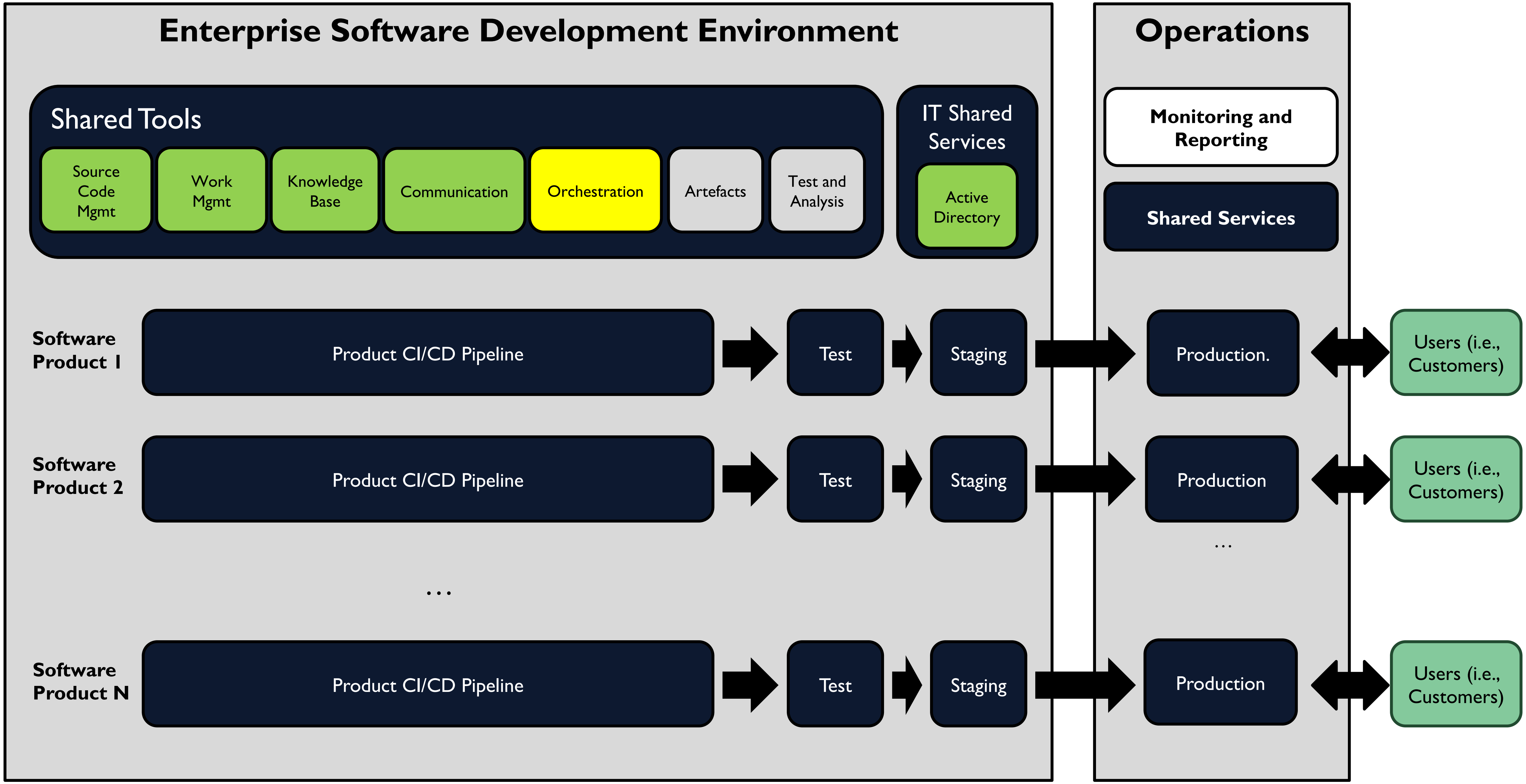
- What are the three practice areas of DevOps?
- What are some characteristics of a good Continuous Integration Pipeline?
- How often should your Continuous Integration Pipeline run (ideally)?
- What types of files should you put in your Source Code Repository?

WARM UP QUESTIONS

- What is a Stakeholder?
- Who would be examples of Stakeholders for our Enterprise Development Environment?

QUIZ 3

- On Stakeholders, DevOps and Continuous Integration
- On the Learning Hub (aka D2L)
- Your have 15 minutes to complete it



Enterprise Software Development Environment
Operations
Shared Tools
Source Code Mgmt
Work Mgmt
Knowledge Base
Communication
Orchestration
Artefacts
Test and Analysis
IT Shared Services
Active Directory
Monitoring and Reporting
Shared Services
Software Product 1
Product CI/CD 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, Assignment 1 Due
6	• CI Pipelines – Shared Libraries	Lab 6, Quiz 5
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	Lab 8, Quiz 6 (Set C), Quiz 7
10	• Spring Break	
11	• CI Pipelines – Artifact Management (Java)	Lab 9, Quiz 8, Assignment 2 Due
12	• Continuous Delivery (CD) • CD Pipelines - Containerization	Lab 10, Quiz 9
13	• CD Pipelines – Deployment • Developer Workflows *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**	

TOOL EVALUATIONS – WHY?

You may <u>need</u> a tool evaluation:

- When you don’t know what tools are available for a particular capability
- When you have competing opinions on which tool to use
- When you need to do due-diligence on a selection to justify the choice or cost (i.e., there is a large cost – either for the tool itself or for transitioning the team to the new tool).

You may <u>not need</u> to do a tool evaluation:

- When the tool has already been selected for you (i.e., a contractual requirement)
- When there is an “industry standard” tool
- When most of your team is already familiar with and there is consensus on a particular tool
- When your team is small and can come to a consensus within the team

A PROCESS FOR TOOLS EVALUATION AND SELECTION

1. Identify the Stakeholders
2. Gather Requirements
3. Research and Comparison
4. Prototype/Proof of Concept
5. Make a Recommendation

You will be doing this for your Assignments 1 and 2

IDENTIFY THE STAKEHOLDERS

*A stakeholder in the evaluation and selection of a tool for the Enterprise Development Environment is and individual or team that has an interest in the use, deployment, operations, maintenance or costs of the tool.**

Example Stakeholders	Description
Project Manager	Responsible for the scope and budget of a IT or software development effort.
Development Manager/Team Lead	Responsible for a group of software developers.
Software Development Team	Responsible for the development of infrastructure and features for a software product. They will be using the tools in the Enterprise Development Environment.
IT Team	Responsible for providing and supporting the IT infrastructure for the organization, including the Enterprise Development Environment.

* *Adapted from a definition of stakeholder from Software Systems Architecture (Rozanski/Woods)*

GATHER REQUIREMENTS - TYPES

Mandatory vs. Optional Requirements

Type	Description
Mandatory	Necessary for the tool to provided the intended capability.
Optional	Nice to have, but not absolutely necessary.

Functional vs. Non-Functional Requirements

Type	Description
Functional	Features of a Product
Non-Functional	Qualities of a Product (i.e., Scalability, Security, Supportability, Maintainability)

GATHER REQUIREMENTS - STAKEHOLDERS

Make sure you consider the concerns of each of the stakeholders (examples below). Those will form the requirements for your evaluation.

Example Stakeholders	Functional Concerns	Non Functional Concerns
Project Manager	Reporting	Price and overall cost (license fees, per user fees, hosting fees, training, upfront vs. ongoing). Scalability/Availability
Software Manager	Reporting User Management Migration from existing tools Integration with other tools	Efficiency
SW Development Team	Availability of specific features	Usability Efficiency
IT Team	Logging Diagnostics User Management/SSO Integration	Supportability Maintainability Security

RESEARCH AND COMPARISON

- Identify which tools are in the market.
- Start with online product information for research.
- Leverage vendor representatives where possible. They can provide product demos and answer specific questions.
- Create a comparison table to show which products meet or do not meet the requirements.

	Product 1	**Product 2**	**Product 3**
Migration from Existing Tool (Required)	✓	✓	✓ Requires 3rd Party Tool
Works with Datasets > 1GB (Optional)	✓	✓	✗ Max Supported 500MB
Pricing (20 users)	$200/month ($10 per user)	$400/month (unlimited users)	Free (Open-Source)

SPECIAL CONSIDERATIONS

- Pricing and Cost
- Cloud vs. On Premise
- Build vs. Buy
- Multiple Products vs. All In One
- Familiarity
- Team Dynamics and Politics

SHORTLISTING – ASSIGNMENT 1

- Use your comparison table to create a shortlist of product(s) to evaluate further
  - Use compliance with your mandatory requirements as the primary criteria
  - Also consider:
    - Optional requirements
    - Pricing and Cost
    - Vendor history and product usage
- At this point, you may be ready to make a recommendation or may require more information

PROTOTYPE/PROOF OF CONCEPT – ASSIGNMENT 2

- Try before you buy
- Identify a sample use case for the product and test it out
  - Leverage free trials
  - Some vendors may support a "proof of concept" providing short-term product licenses and dedicated support
- Involve your stakeholders, especially those who will be actively working with the product
- Make sure it actually meets your requirements. The use case should be realistic.

MAKING A RECOMMENDATION – ASSIGNMENT 2

- Present your evaluation to the stakeholders – it may be a written report or a presentation
  - Include your comparison table
  - Include the results of any prototype or proof of concept evaluation
- Show that your have considered the stakeholder inputs through the requirements
- Be objective and present the facts from your research and product evaluation
  - Backup your recommendation through quantitative and qualitative assessment
  - Quantitative – i.e., against the requirements
  - Qualitative – i.e., hands on usage
- Decision could be political, try not to get emotionally involved

TOOL EVALUATIONS - BEST PRACTICES

- Involve your stakeholders early and throughout the process
  - Helps you to gather the real requirements and avoid surprises at the end
  - May make the decision making process easier
- Make sure you also know your constraints, such as:
  - Budget – both up front and ongoing (i.e., yearly)
  - Supported IT Environment – OSes, Databases, On-Premise/Cloud, Etc.
  - Regulations/Standards – Security Audits, Data Privacy, Access Controls
- While free open-source products are popular, larger enterprises often need extra features and support not available in the free versions. Don't exclude paid products or plans by default.
- If you or your team are not familiar with a product, do not rely purely on product research. Make sure you talk to the product vendor and conduct a hands on evaluation before you commit.

DEVOPS

Development and Operations Engineers participating in the entire software lifecycle, from design through development and deployment to production support.

This is in contrast to Developers only participating in design and development and Operations only participating in deployment and production support. This creates a strong handover that often leads to friction and inefficiencies.

Other characteristics of DevOps include:

- Operations Staff making use of the same tools as developers for the systems work (i.e., source code management, infrastructure as code, automation)
- It aligns with agile values, principles, methods and practices.

DEVOPS – PRACTICE AREAS

Three primary practice areas:

- **Infrastructure Automation** – Define systems, OS configuration and Application deployments in code (i.e, your Provisioning Couse).

- **Continuous Delivery** – Build, test and deploys the Applications your build in a fast and automated manner (Continuous Integration, Continuous Delivery and Continuous Deployment).

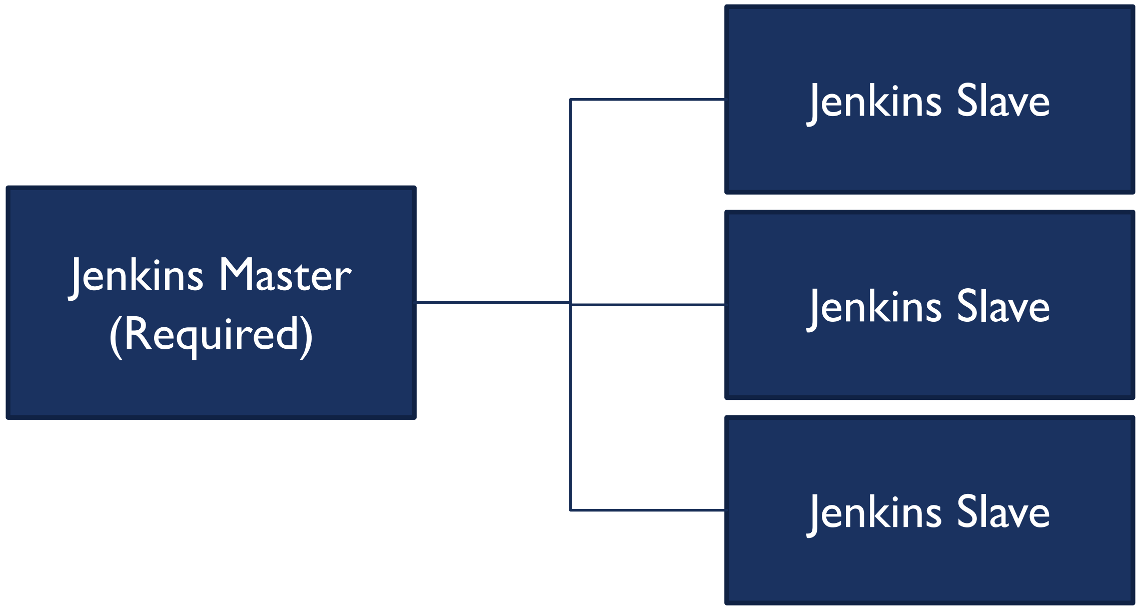
- **Site Reliability Engineering** – Operating, monitoring and orchestration of your systems.

CONTINUOUS INTEGRATION

- **Continuous Integration (CI)** – Every time code is changed in the source code repository (or at least daily) the code is checked out to a build machine where it is built and unit tests are run.
- **Continuous Delivery (CD)** – The built software artefacts are automatically deployed to internal environments (dev/test) where further automated or manual tests are run.
- **Continuous Deployment** – The built software artefacts are automatically deployed to production if the automated steps from the previous CI/CD steps all pass.

CI TOOL - JENKINS

- Open-Source automation tool. Written in Java and support plugin for CI purposes.
  - Integrates with Source Code Management Tools (Git, SVN), Test Tools, Deployment Tools, Etc.
- Provides Orchestration of CI/CD Pipelines



Can optionally have 1 or more Jenkins Slaves to distribute build jobs and allows builds on machines with different configurations (i.e., Java installed, Python installed)

NEW LAB REQUIREMENTS

- **REQ1130** – The Enterprise Development Environment shall have an Orchestration capability to enable CI/CD Pipelines. The Orchestration tools will be Jenkins.
- **REQ1140** – The Orchestration capability shall integrate with the Source Code Management Capability. Jenkins will pull source code from GitLab.
- **REQ1100** – The Enterprise Development Environment shall provide access to applications on the web through a single web application server acting as a reverse proxy. The reverse proxy will be implemented using Apache2.
- Note: We will add Jenkins to the reverse proxy.

NEW LAB REQUIREMENTS

- **REQ1130** – The Enterprise Development Environment shall have an Orchestration capability to enable CI/CD Pipelines. The Orchestration tools will be Jenkins.

We will just deploy a Jenkins Master today as a container using Docker.

DOCKER

- Docker is a container engine on top of the Linux Kernel
- A container is a standardized unit of software, including all dependencies required by the software
- Allows us to automate application deployment on the container and automate the startup and shutdown of those containers in a runtime environment

There are standard Docker images you can use and extend (i.e., from DockerHub). We will be doing this for our Jenkins install today.

In the upcoming weeks we will also be building and deploying Docker images as part of our CI/CD pipelines.

DOCKER

Dockerfile

```
FROM jenkins/jenkins
LABEL maintainer="mmulder10@bcit.ca"
USER root
RUN mkdir /var/log/jenkins
RUN mkdir /var/cache/jenkins
RUN chown -R jenkins:jenkins /var/log/jenkins
RUN chown -R jenkins:jenkins /var/cache/jenkins
USER jenkins
ENV JAVA_OPTS="-Xmx4096m"
ENV JENKINS_OPTS="--handlerCountMax=300 --
logfile=/var/log/jenkins/jenkins.log --
webroot=/var/cache/jenkins/war --prefix=/jenkins"
```

Docker Commands

docker build – builds a Docker container image from a dockerfile

docker run – runs a Docker container

docker stop – stops a running Docker container

docker ps – shows running Docker containers

docker exec – runs a linux command on a running Docker container

NEW LAB REQUIREMENTS

- **REQ1140** – The Orchestration capability shall integrate with the Source Code Management Capability. Jenkins will pull source code from GitLab.

You will be adding a new Project with some Python code to your GitLab installation.

This code will be pulled by a Jenkins job and the unit tests run.

We will just be creating a Freestyle Jenkins job today. Next week we will create a Pipeline job.

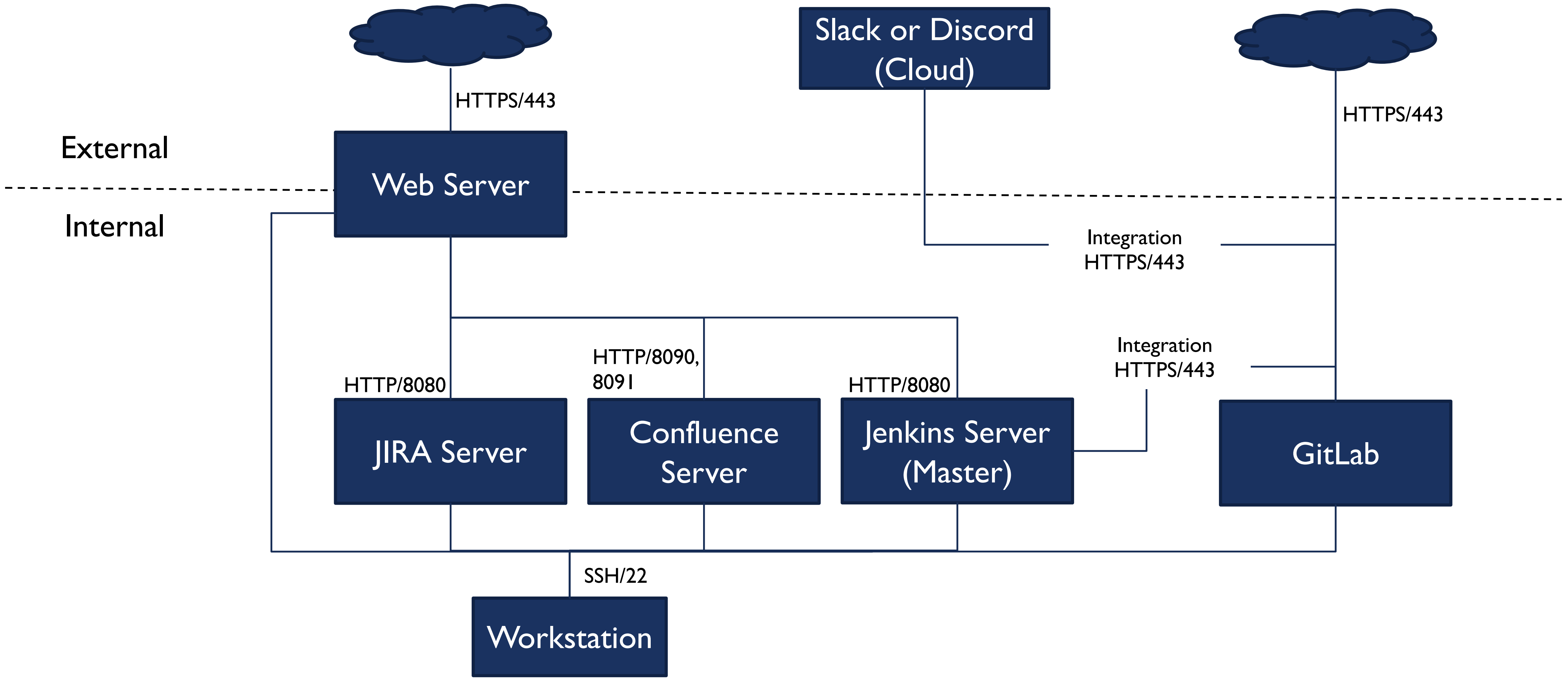
NEW LAB REQUIREMENTS

- **REQ1100** – The Enterprise Development Environment shall provide access to applications on the web through a single web application server acting as a reverse proxy. The reverse proxy will be implemented using Apache2.
- Note: We will add Jenkins to the reverse proxy.

Jenkins runs by default on Port 8080. You will be adding Jenkins to your reverse proxy so it can be accessed on:

https://<Your Apache DNS Name/jenkins

YOUR ENTERPRISE DEVELOPMENT ENVIRONMENT (SO FAR)



TODAY’S LAB

1. Demo Lab 3 Before the End of Class
2. Start on Lab 4
    1. You will do this together with your partner
    2. First part is to upgrade your GitLab installation
    3. Demo is due in class next week.
3. Next week we will be creating Jenkins Pipeline jobs