ENTERPRISE SYSTEMS INTEGRATION

ACIT4850 – WINTER 2024

COURSE OVERVIEW

This course looks at building and maintaining an environment for Enterprise Software Development.

Students will learn about the components that make up this environment, including the following topics:

- Software Source Code Management
- Work Management and Issue Tracking
- Knowledge Sharing and Communication
- Continuous Integration (CI)
- Automated Test
- Static Code Analysis
- Artifact Management
- Continuous Delivery (CD)

Students will incrementally deploy, configure and integrate the tools for an enterprise software development environment and exercise common software development and test workflows within this environment.

Students will also conduct tool assessments and trade-off analysis between cloud and on-premise tools and environments.

LEARNING OUTCOMES

1. Identify the characteristics and components of an environment for enterprise software development
2. Deploy and configure the tools needed to support an enterprise software development environment
3. Create and configure Continuous Integration pipelines for software builds
4. Create and configure Continuous Delivery pipelines for automated deployments
5. Setup software development workflows within the enterprise software development environment
6. Conduct tool assessments and trade-off analysis

LEARNING RESOURCES

Textbook

- Online resources and papers will be provided weekly. No textbook is required.

Software

- Access to a Cloud Environment for Class Labs. We will be using Microsoft Azure as the cloud services provider.
  - They do offer $100 in credits to students
  - You will likely need to pay for some Azure services (when your credit runs out)
- Purchase of some software licenses may be required.

EVALUATION CRITERIA

Quizzes	15%	There will be a quiz at the beginning of each class (except the first and last)
Project Labs	35%	The full enterprise software development environment will be incrementally built through the labs. Includes a participation component.
Assignments	15%	There will be 3 assignments, two individual and one with your partner
Final Exam	35%	

Passing Criteria:

- Students must complete a minimum of 80% of the Project Labs.
- Students must achieve a minimum of 50% on the combination of the Quizzes and Final Exam (i.e., 25/50).
- Students must achieve an overall grade of at least 50% in the course.

Quizzes will **only be done in-class only.** There are no late or make-up quizzes, but the lowest quiz of the term will be dropped.

There is a 20% penalty per week for late lab submissions, up to a maximum of 2 weeks.

COURSE COMPONENTS

- Quizzes – Weekly based on a reading published after the previous class
- Project Labs – Deploy, configure and use an enterprise development environment.
  - Done with a partner in most cases. Requires contribution by both students.
  - Labs incrementally build on each other. You typically can't miss a lab without making it up in some way.
  - Each lab is due and demoed by the end of the next class.
- Assignments – Tool Evaluations, Developer Workflows
  - Two are individually, one is with a partner
- Final Exam – Concepts and Practical

CLASS STRUCTURE

Pre-reading Before Class – On the week's tool/concept

Classroom (30-60 minutes):

- Quick Review – On the reading
- Quiz – On Learning Hub
- Discussion/demo on the week's lab and/or concept

Lab (remainder of class time)

- Demo of previous week's completed lab
- Work on new lab

LABS

You will receive full marks if you meet all of the requirements (i.e., the technical requirements).

If you do not meet all the requirements, you can re-demo the lab the following week but you will **lose 20% per week for a maximum of 2 late weeks**

Participation will also be a component of your lab mark based on lab demos (attendance, conducting the demo, answering questions). **Both partners in a group must attend the lab demo otherwise I won't mark it.**

LET'S GET STARTED…

DEFINITIONS

Enterprise

- A business or company

System

- In this context, software platforms and applications

Integration

- Connecting the systems together to share and exchange data

DEFINITIONS

Enterprise System

- Software applications used by companies to perform their day to day operations.
- Examples include: HR systems, Payroll applications, CRM, CMS, customer order entry

Enterprise System Integration

- Making the software used within a company share data and exchange data so that processes are more seamless and efficient across the organization
- For example, if a new employee is hired and added to the HR system they should also be automatically added to the Payroll application.

FOR THIS COURSE

Enterprise

- A company that develops software products. Their new software product is a web application used sell their products to other businesses.

Enterprise Systems

- In this case, the Enterprise Systems are the tools that support the definition, development, test and deployment of their software product.

Enterprise System Integration

- The Enterprise Systems are configured to work together to facilitate the efficient definition, development, test and deployment of the software product.

THE SCENARIO

Me

- I am a Software Architect at your company but have been assigned as the Product Owner for a new initiative to prototype an integrated Enterprise Software Development environment.
- I will be providing you with technology selections and requirements for the environment through your weekly labs. You will also sell-off your progress against those requirements to me on a weekly basis.

You

- You are a newly hired IT staff member at your company.
- You and a co-worker have been tasked to prototype the Enterprise Software Development environment.
- In some cases you will need to make technology recommendations/selections and produce documentation on the environment.

THE END USERS

Software Developers

- There will be multiple software products under development at various stages of the software lifecycle, each with a separate software development team (say 3-5 teams) of 3-7 developers (i.e., a Scrum team)
- They want to encourage more collaboration, knowledge transfer and sharing of common code across teams

Project Managers and Product Owners

- There will be one Project Manager and one Product Owner per Software Development Team. They may work across multiple teams.
- Project Managers want more visibility into the progress of the software development teams (i.e., monitoring and control)
- Product Owners need to be able to manage the use cases, user stories and requirements for a software product

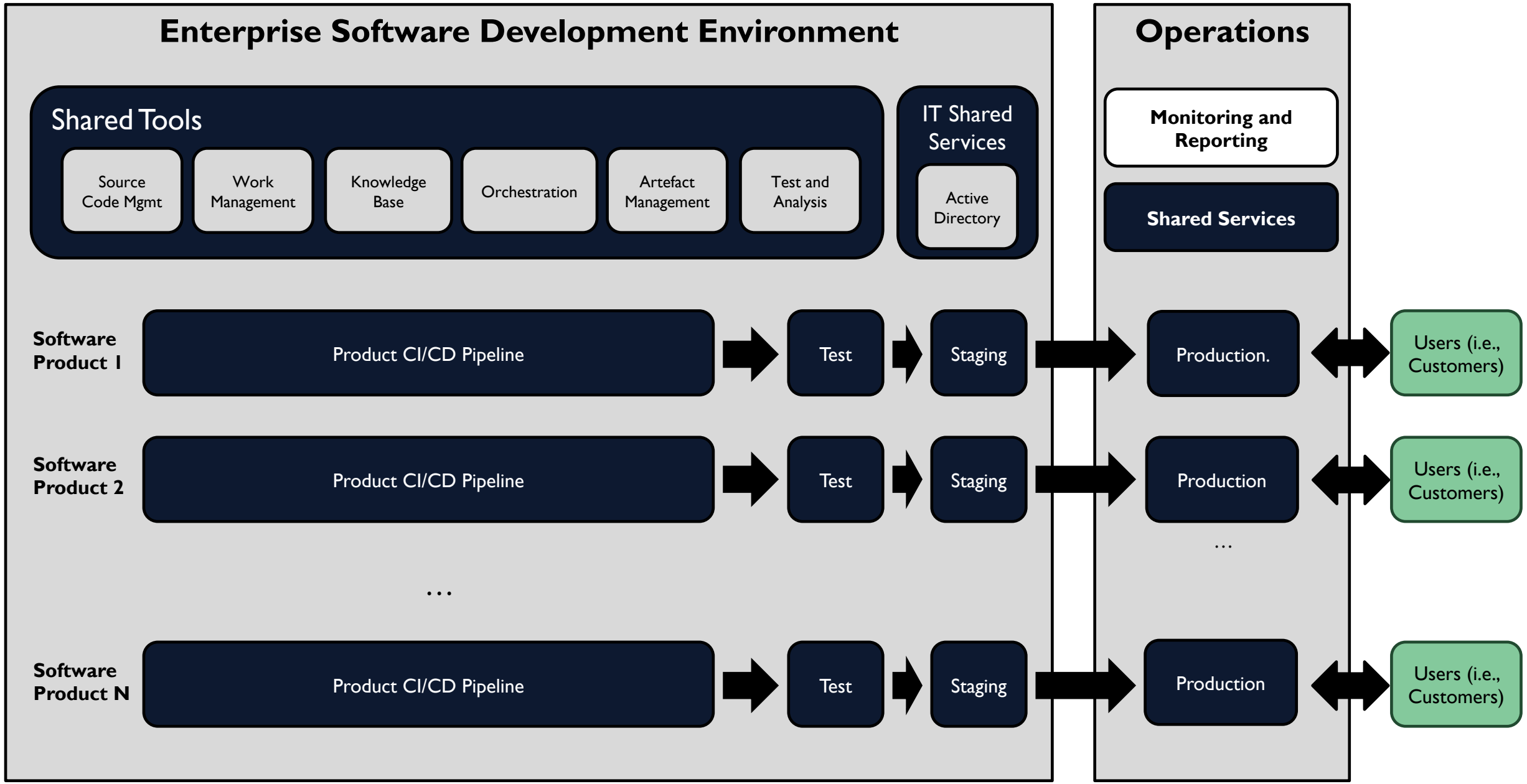
THE END USERS

Test Team

- There will be 1-2 testers per Software Development Team. They may work across multiple teams.
- They want to have regular access to test environments with working software for verification and validation

Operations Team

- There will be 1-2 operations staff per Software Product. They may work across multiple products.
- They want operational concerns (i.e., deployment, maintainability) considered earlier in the software development lifecycle
- They want quick turn around of issues found in production


Enterprise Software Development Environment
Shared Tools
Source Code Mgmt
Work Management
Knowledge Base
Orchestration
Artefact Management
Test and Analysis
IT Shared Services
Active Directory
Operations
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)

CHECKIN SURVEY

Identify which specific tools you have used in the past (school/work) for the following:

- Source Code Management – Central repository of latest and historic source code (i.e., Git, Subversion tools)
- Work Management – Manages tasks and issues (i.e., Kanban/Scrum board, issue tracker)
- Knowledge Base – Stores team or project knowledge on a software product (i.e., wiki, document management)
- Orchestration – Initiates CI/CD pipelines (i.e., build/test of software)
- Source Code Quality – Analyzes code for common coding issues and conventions
- Code Security Analysis – Analyzes code for common security issues
- Automated Test – Tests the software in an automated manner as unit, integration or user interface tests
- Centralized Authentication – To enable single sign-on

Spend <5 minutes and complete the Checkin Survey
(Activities -> Surveys) to answer the above

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, Assignment 1 Due
5	• CI Pipelines – Python	Lab 5, Quiz 4
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	• CI Pipelines – Artifact Management (Java)	Lab 9, Quiz 8, Assignment 2 Due
11	• Continuous Delivery (CD) • CD Pipelines - Containerization	Lab 10, Quiz 9
12	• CD Pipelines – Deployment Part 1	Lab 11, Quiz 10
	• CD Pipelines – Deployment Part 2 *Note: At home lab for Monday set*	Lab 12, Quiz 11 (Sets A and B)
14	• Microservices Pipelines • Final Exam Preview	Quiz 11 (Set C), Assignment 3 Due
15	**Final Exam**	

This is a Roadmap so it will be refined as we go.

ASIDE - SYSTEM REQUIREMENTS

- Formal representation of the requirements of a system, part of the System Engineering Body of Knowledge (SEBOK).
- Typically leverages specific terms:
  - **shall**: Requirement is contractually binding, meaning it must be implemented and verified
  - **will**: Statement of fact. Not subject to verification.
  - **should**: Goals, non-mandatory vision. Typically a goal which must be addressed by the design team but is not formally verified.
- Note: Do not use **must** as no one has defined how must is different from shall. Also, shall has held up in court, must has not.
- The above is based on ISO TR 10176

ASIDE - SYSTEM REQUIREMENTS - EXAMPLES

- **Shall** Requirement
  - The system shall process updates from the data source within 6 seconds from initial receipt.
- **Will** Requirement (describing another system)
  - The data source will provide updates every 10 seconds in Json format.
- **Should** Requirement (design requirement)
  - The system should not prevent users from carrying out other activities while it is processing updates from the data source.

- Notes:
  - Will and should requirements are typically to provide context to the audience for the requirements (i.e., customers, developers).
  - Often notes will be added to requirements to provide further context.

FIRST SET OF REQUIREMENTS

- **REQ1010** - The Enterprise Development Environment shall be prototyped on Microsoft Azure cloud infrastructure. Note: All technical choices should allow for either on-premise or cloud infrastructure for the production implementation.
- **REQ1020** – The Enterprise Development Environment shall have a Source Code Management capability. The Source Code Management tool will be GitLab. Note: The GitLab Community Edition (free) will be sufficient for the prototype environment.
- **REQ1030** - The Source Code Management tool shall be integrated with the single sign-on services provided by the Microsoft Azure cloud infrastructure. Note: The oauth integration from Azure Active Directory (AD) will be used for the prototype.
- **REQ1040** – The Source Code Management tool shall allow software development teams to create source code repositories and logically group them. Note: Source code repositories are Projects in GitLab and groupings of repositories are Groups in GitLab. Logical groupings may be by software product and common infratructure.
- **SEC1010 –** All password credentials will be stored in a secure place (i.e., password safe).
- **SEC1020** – All web applications and API endpoints shall be encrypted (i.e., https endpoints). Note: Self-signed certificates are sufficient for the prototype environment.

ENTERPRISE DEVELOPMENT ENVIRONMENT INFRASTRUCTURE

REQ1010 - The Enterprise Development Environment shall be prototyped on Microsoft Azure cloud infrastructure. Note: All technical choices should allow for either on-premise or cloud infrastructure for the production implementation.

What do we mean by On-Premise? Cloud?

- On Premise – Your own infrastructure controlled by your company. Could be bare metal or virtual. For software, on-premise means it is deployed to on-premise infrastructure.
  - Usually more of an upfront expense (i.e., capital expense)
- Cloud – On someone else's infrastructure that you pay to use on an incremental basis (i.e., pay incrementally).
  - Usually more of an ongoing expense (i.e., operational expense)

SOURCE CODE MANAGEMENT

REQ1020 – The Enterprise Development Environment shall have a Source Code Management capability. The Source Code Management tool will be GitLab. Note: The GitLab Community Edition (free) will be sufficient for the prototype environment.

REQ1040 – The Source Code Management tool shall allow software development teams to create source code repositories and logically group them. Note: Source code repositories are Projects in GitLab and groupings of repositories are Groups in GitLab. Logical groupings may be by software product and common infratructure.

- What are other source code management tools (other than Git) are out there?
- Who are GitLab's competitors?
- What criteria would you use to select a Git based Source Code Management Tool for our prototype?
- Why might we want to have our Source Code Management Tool on-premise rather than using a hosted SaaS?

SOURCE CODE MANAGEMENT - EXERCISE

Evaluate the top 3 Git Based Source Code Management Tools based on criteria that may be important for our Enterprise Development Environment.

Source Code Management Tools:

- GitHub
- GitLab
- Bitbucket

Criteria Examples:

- Features
- Price/Cost
- Support
- Outstanding Security Issues
- Overall Usage
- Familiarity
- …

Does anyone have a strong personal opinion on which Git Based Source Code Management Tool to use?

SINGLE SIGN ON

REQ1030 - The Source Code Management tool shall be integrated with the single sign-on services provided by the Microsoft Azure cloud infrastructure. Note: The oauth integration from Azure Active Directory (AD) will be used for the prototype.

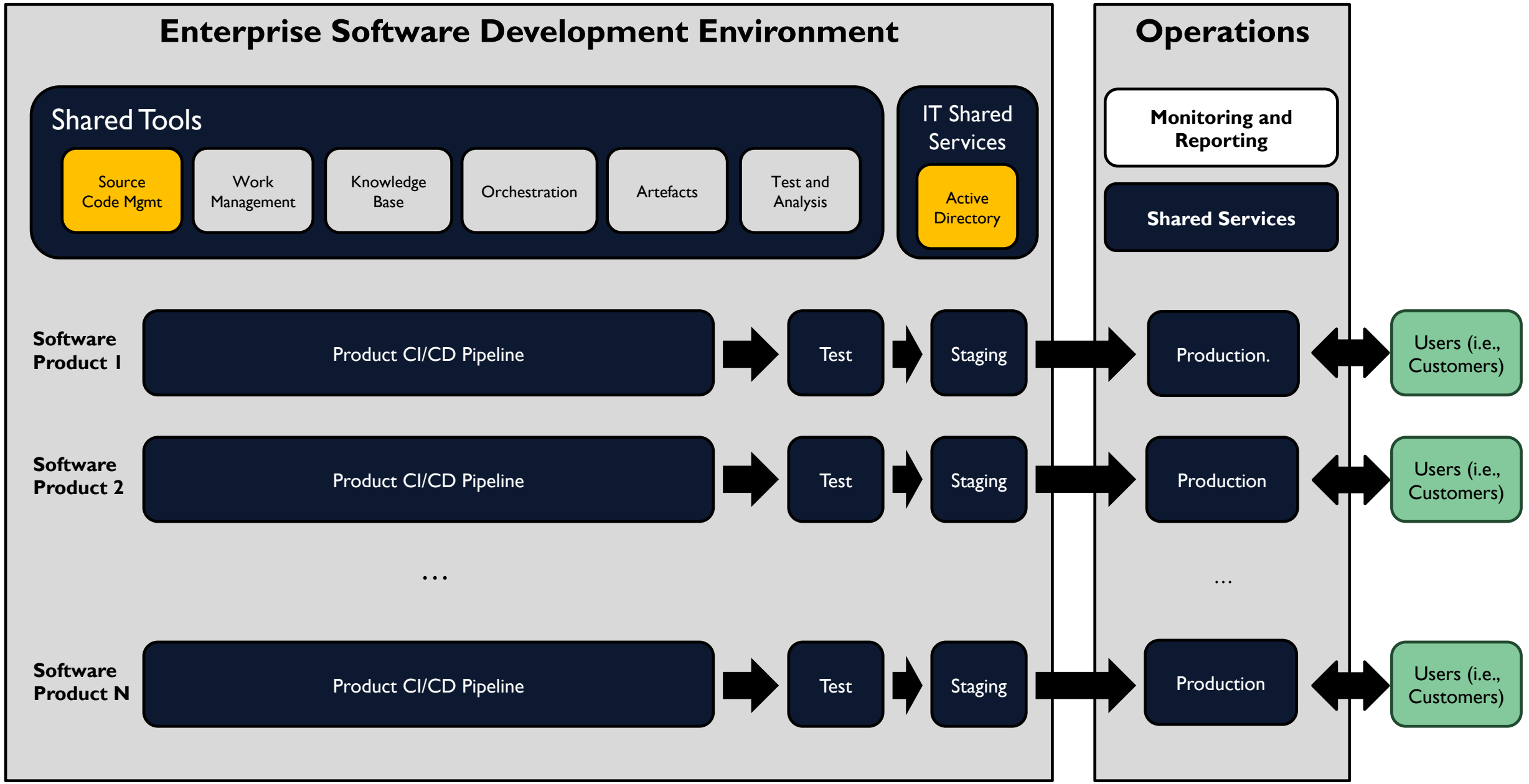
- Have you used ActiveDirectory, LDAP or OAuth before?
- Why is this requirement important in an Enterprise Environment?

SECURITY

SEC1010 – All password credentials will be stored in a secure place (i.e., password safe).

SEC1020 – All web applications and API endpoints shall be encrypted (i.e., https endpoints). Note: Self-signed certificates are sufficient for the prototype environment.

- What other security requirements do you think would be important for our Enterprise Development Environment?
- Would on-premise vs. cloud affect your opinion?


Enterprise Software Development Environment
Shared Tools
Source Code Mgmt
Work Management
Knowledge Base
Orchestration
Artefacts
Test and Analysis
IT Shared Services
Active Directory
Operations
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)

LAB 1

- You may work with a partner. This should be the **same partner for the duration of the course,** so choose wisely.
- Sign-up (with your partner) for a Group on the course site on the Learning Hub (Course Tools -> Groups). Choose a group with your Set in the name and isn't already filled.
- The lab must be demoed by end of next class. Both group members must participate.
- You are responsible for managing the costs of the Azure resources you use. **Make sure you stop (but don't delete) anything you are not using (i.e., Virtual Machines).**

Good Luck!
If you run into problems in the lab let me know and share solutions to issues that you've solved with your classmates.