ENTERPRISE SYSTEMS INTEGRATION

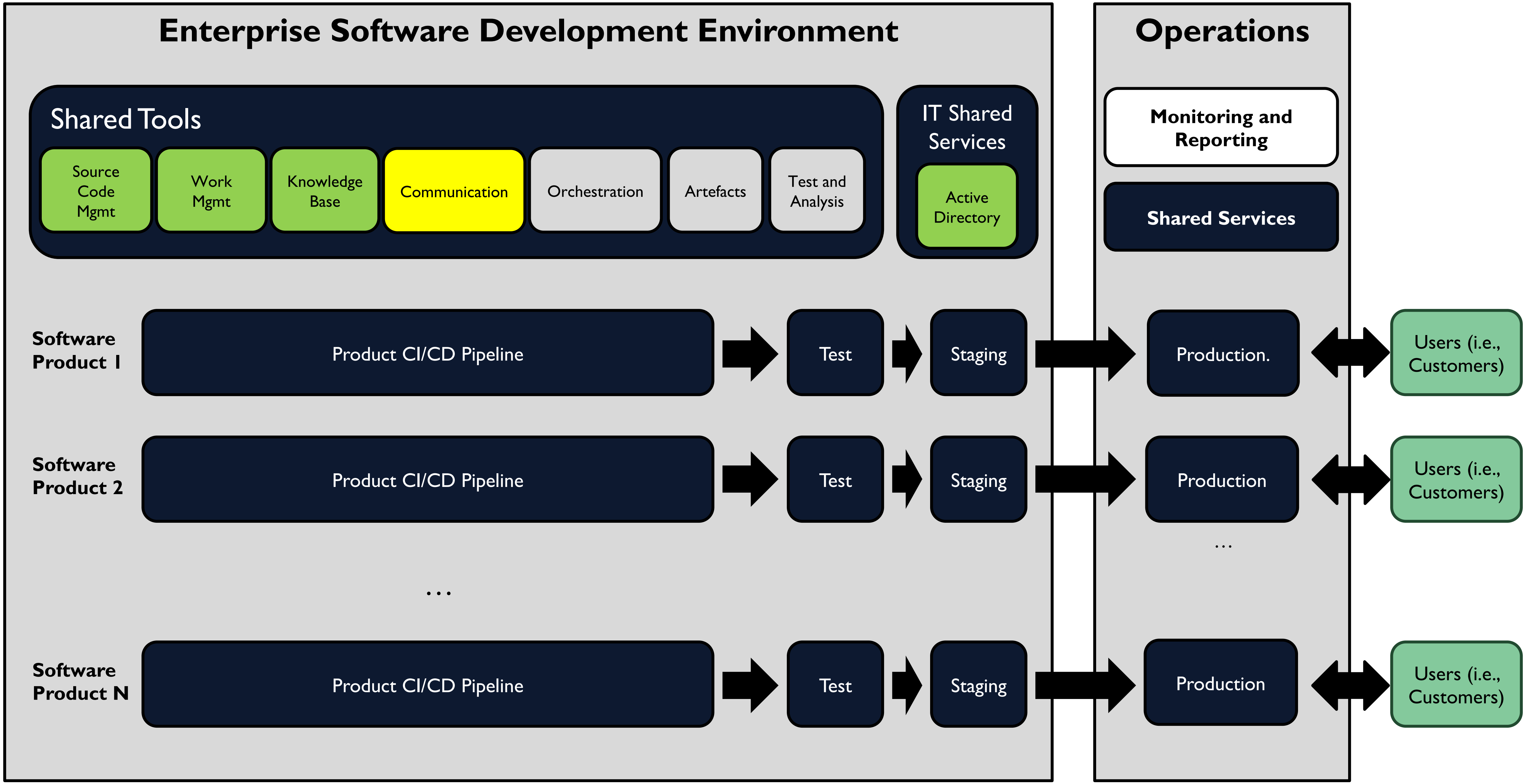
ACIT4850 – WINTER 2024

AGENDA

- Quick Review
- Quiz 2 on D2L
- Non-Functional Requirements
  - Usability
  - Recoverability
  - Pricing
  - Security
- Production Environment
- Governance
- Lab Requirements
- Lab
  - Demos of Lab 2
  - Start on Lab 3

QUIZ 2

- On Functional and Non-Functional Requirements
- On the Learning Hub (aka D2L)
- Open Book
- Your have 15 minutes to complete it



Enterprise Software Development Environment

Shared Tools

Source Code Mgmt

Work Mgmt

Knowledge Base

Communication

Orchestration

Artefacts

Test and Analysis

IT Shared Services

Active Directory

Software Product 1

Product CI/CD Pipeline

Test

Staging

Software Product 2

Product CI/CD Pipeline

Test

Staging

. . .

Software Product N

Product CI/CD Pipeline

Test

Staging

Operations

Monitoring and Reporting

Shared Services

Production.

Production

. . .

Production

Users (i.e., Customers)

Users (i.e., Customers)

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**	

FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

Functional Requirement

- Specifies something the system should do.
- Describes the behavior or features of the system.

Non-Functional Requirement

- Describe how the system works.
- Specifies **how the system should behave** and that it is a constraint upon the systems behaviour.
- Quality attributes of a system.

REVIEW THIS SCENARIO

The Software Development Team at the company you work for currently uses Subversion for their source code management. They have been asking to move to a Git based source code management tool and your supervisor has tasked you with researching and recommending a solution from the following options: GitHub, GitLab or Bitbucket.

The company is willing to pay a license fee for the product and would prefer an option that includes support and a good developer experience. They have a maximum budget of $500/developer per year.

Because they already have all their code repositories in Subversion, there needs to be an efficient solution for migrating existing source code from Subversion to Git that maintains the history of all the revisions to the source code currently in the Subversion repositories.

The Software Development Team has approximately 30 developers and 4 unique products. Each product has multiple source code repositories ranging in size from 100K lines of code to 4 million lines of code. The also want a way to organize the source code repositories by product and restrict write permissions based on the product the developer is working on. They are also looking to introduce static source code analysis for code quality and security vulnerabilities in the future so they would prefer a solution that provides those features (preferred) or provides integration with other tools that provide those features.

The company is also looking to expand their product offerings in the next three years with 8 additional products and increasing the size of the development team to over 100 developers working across three different continents (and across the Vancouver, London and Singapore timezones).

What might be a functional requirement from this scenario?

What might be a non-functional requirement from this scenario?

NON-FUNCTIONAL REQUIREMENTS

Characteristics

- Cross-cutting
- Hard to test
- Need to be designed in up-front
- Can be expensive to implement

Also known as NFRs, Service Qualities, Architecturally Significant Requirements or The "ilities"

Categories

- Performance – for example Response Time, Throughput, Utilization, Static Volumetric
- Scalability
- Capacity
- Availability
- Reliability
- Recoverability
- Maintainability
- Serviceability
- Security
- Regulatory
- Manageability
- Environmental
- Data Integrity
- Usability
- Interoperability

USABILITY

What Usability Requirements would be applicable to our Enterprise Development Environment?

- Single Sign-on
- Browser support (should support the default browser at the company)
- Consistent/Easy to Remember URLs
- Accessibility – Do the tools support screen-readers for those with visual impairments?

RECOVERABILITY

What Recoverability Requirements would be applicable to our Enterprise Development Environment?

- If a VM or server goes down, can it be recovered within a given period (i.e., a few minutes, a few hours)
- If a database is corrupted or goes down, can it be recovered within a given period (i.e., a few minutes, a few hours)
- If a disaster occurs at the company or cloud provider’s premise, can the systems be redeployed to an alternate site within a given period (i.e., minutes, hours, days)

PRICING

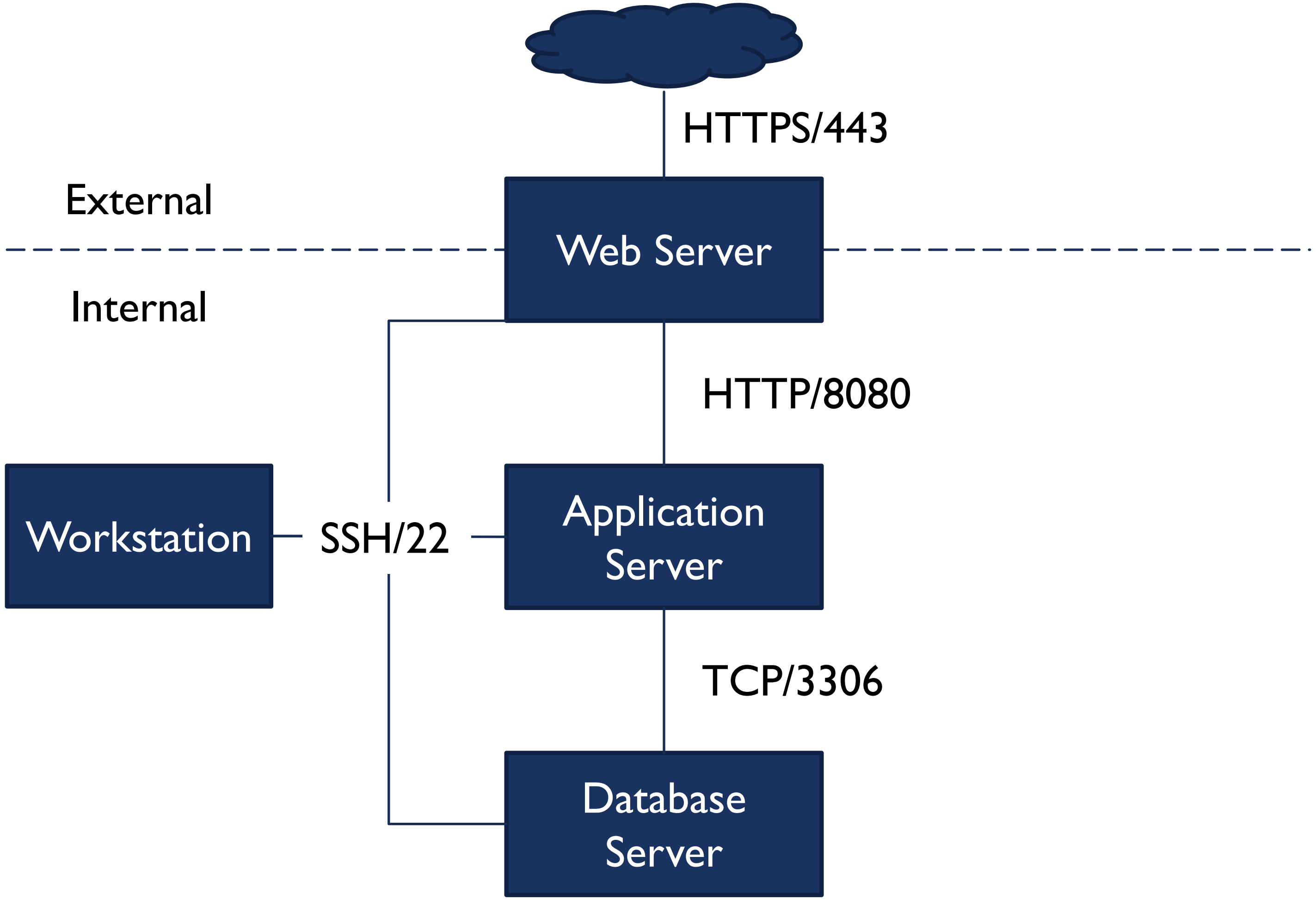
- This would be considered a non-functional requirement
- This is especially important for tool selections as it can impose a constraint on your selection
- You should try to understand:
  - What is the budget available? Sometimes the better question: Is there a budget available?
  - Is the budget one-time (capital) or monthly/yearly (operational)
  - Sometimes to may have to convince your stakeholders to allocate a budget, especially if there are no sufficient "free" choices

SECURITY

What Security Requirements would be applicable to our Enterprise Development Environment?

- Layered security – separate web server, application server and database server
- Networking – only the required ports and protocols are open on any given server
- Non-Standard Configurations (i.e., Application Hardening) – Change the default ports, users/passwords in the system (security by obscurity)
- Encryption – All data in transit encrypted using SSL
- VPN – If internal facing only, require that users must login to a VPN before they can access the applications

SECURITY – LAYERED SECURITY



Layered Security Characteristics

- Limited external footprint
- Only the required protocols/ports are open between servers and workstations
- Separate servers each with a single purpose (i.e., the layers):
  - Web Server – External Facing
  - Application Server – Internal
  - Database Server – Internal
- Limited remote access through SSH (specific IPs only)

PRODUCTION AND STAGING ENVIRONMENTS

Production Environment – Where your applications are deployed for use by the end users

Staging Environment – Where you test deployments of updated applications before deployment to production

- As close to production as possible
- You may have limited time to deploy to production (specific maintenance windows) so you want to make sure you work out any issues early

GOVERNANCE

With respect to our Enterprise Development Environment, it is the rules around which the shared tools and services are used and maintained.

What would be some examples of rules we would want to define?

- Who manages accounts and permissions of the users of the environment?
- What are the maintenance windows when applications and infrastructure can be updated?
- How often are software applications updated (i.e., monthly, quarterly, yearly)?
- Who does a user go to if they have a problem?

QUESTION: IMPROVEMENTS

Given our current applications and their deployments (GitLab, Confluence, JIRA), what improvements might we consider for a production environment with regard to:

- Usability
- Recoverability
- Security

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: This web application server provides a single point of access from the public internet and can be used for SSL termination.

  Note: For this prototype, this web application server will be used for Confluence and JIRA which run by default on non-standard ports.
- **REQ1110** – The Enterprise Development Environment shall have a Communication capability. The Communication tool will be Slack.
- **REQ1120** – The Communication capability shall be capability of receiving automatic notifications from other tools. For this prototype, notifications will be reported from GitLab.
- **SEC1020** – All web applications and API endpoints shall be encrypted (i.e., https endpoints). Note: Self-signed certificates are sufficient for the prototype environment.

REVERSE PROXY

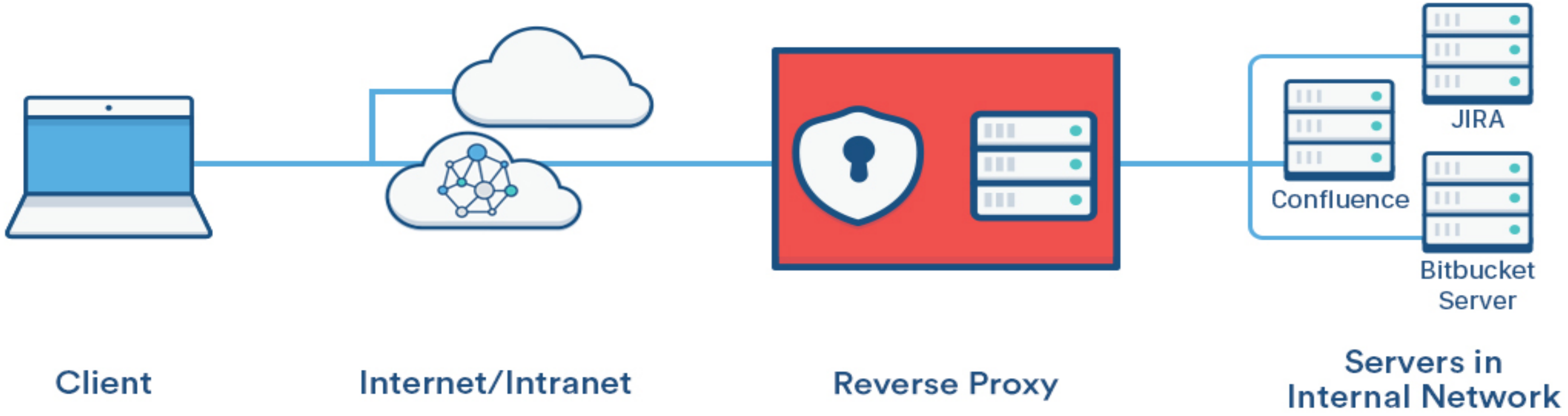
REQ1100 *– The Enterprise Development Environment shall provide access to applications on the web through a single web application server acting as a reverse proxy. The reverse proxy will be implemented using Apache2.*

Note: This web application server provides a single point of access from the public internet and can be used for SSL termination.

Note: For this prototype, this web application server will be used for Confluence and JIRA which run by default on non-standard ports.

Confluence and JIRA were designed to run behind a web application server. We will use Apache2 configured as a reverse proxy for our environment.

REVERSE PROXY



A **reverse proxy** is a type of **proxy** server that retrieves resources on behalf of a client from one or more servers. These resources are then returned to the client, appearing as if they originated from the **proxy** server itself. (*Source: Wikipedia*)

It can:

- *Allow multiple servers to be accessed from a single domain name or IP*
- *Hide the identify of the source servers*
- *Act as an SSL termination point*
- *Limit exposure of multiple servers to the web*
- *Perform load balancing*

We are using it to provide a single domain name for our servers and as a SSL termination point.

COMMUNICATION – SLACK OR DISCORD

- ***REQ1110*** *– The Enterprise Development Environment shall have a Communication capability. The Communication tool will be either Slack or Discord.*
- ***REQ1120*** *– The Communication capability shall be capable of receiving automatic notifications from other tools. For this prototype, notifications will be reported from GitLab.*

Communication within and across development teams is important to foster collaboration and teamwork. Traditionally this has been done with in-person meetings and e-mail. However, real-time communications can be very effective especially for remote teams.

For communications among team members Slack is one of the more populate tools today for software development teams. The integration between GitLab and Slack is very easy to implement and will become more important once we get into software builds and development workflows. **However, since we have larged moved to Discord for many CIT classes you can choose to do this integration instead.**

SECURITY – SELF-SIGNED CERTIFICATE

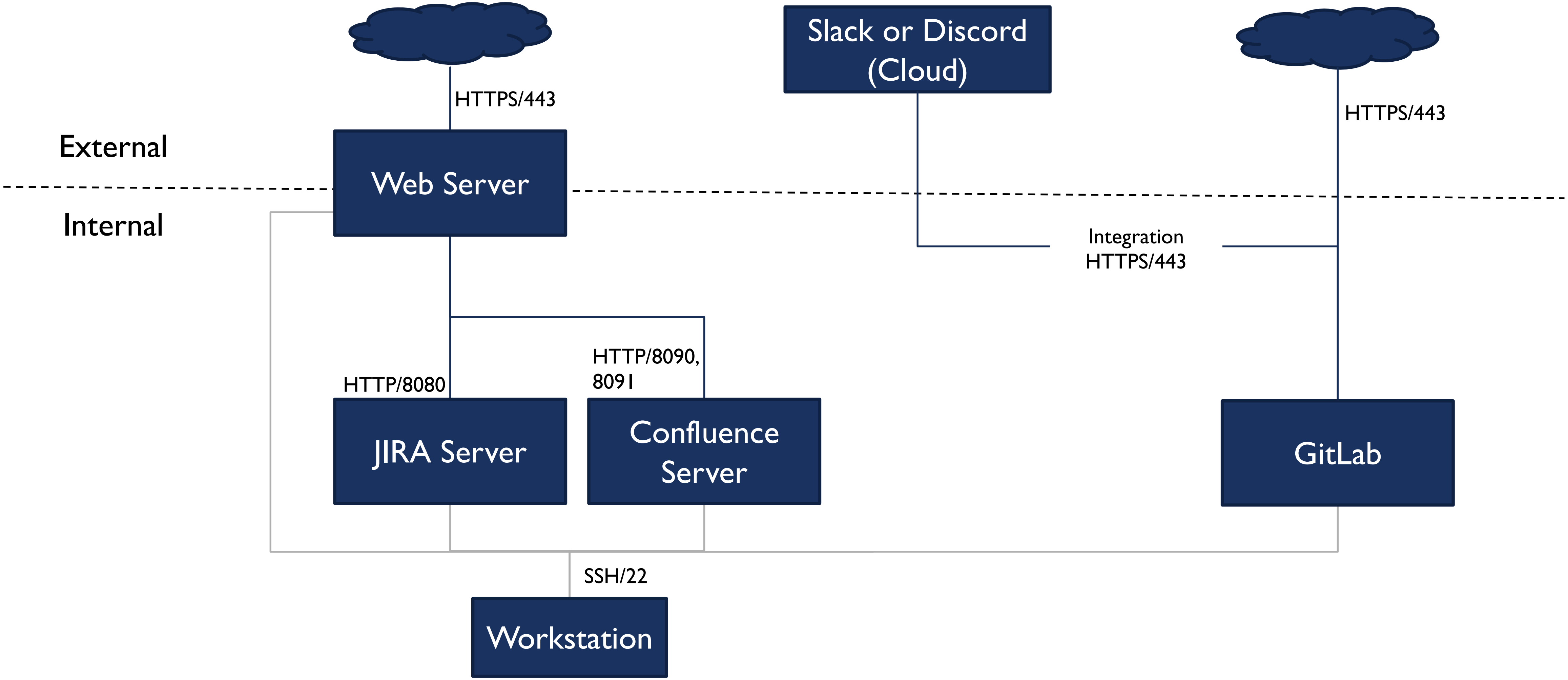
***SEC1020** – All web applications and API endpoints shall be encrypted (i.e., https endpoints). Note: Self-signed certificates are sufficient for the prototype environment.*

You will install a self-signed certificate on your Apache2 reverse proxy so all incoming/outgoing data is encrypted with SSL. We may upgrade to a signed certificate later.

Internally, transmission between the Web Server and Confluence/JIRA Application Servers will be unencrypted as we will consider those internal communication. Better practice would be to have those encrypted as well.

YOUR ENTERPRISE DEVELOPMENT ENVIRONMENT

AFTER TODAY'S LAB



TODAY'S LAB

1. Demo Lab 2 Before the End of Class
2. Start on Lab 3
    1. You will do this together with your partner
    2. Demo is due in class next week (or before).