



Prifysgol Cymru

Y Drindod Dewi Sant

University of Wales

Trinity Saint David

Trawsnewid Addysg, Trawsnewid Bywydau
Transforming Education, Transforming Lives

Application Deployment

Network Programmability
(ACCB5029)

Applied Computing - WISA

Previously,

- ✓ Overview, Linux and Python Skills Review
- ✓ Devnet Developer Environment
- ✓ Software Development & Design
- ✓ Implement software version control using GIT
- ✓ Methods, Functions, Modules and Classes
- ✓ Data Formats
- ✓ APIs, Design and Architecture, REST APIs

This week,

- ✓ Understand Deployment Choices
- ✓ Creating and Deploying Application
- ✓ Continuous Integration/ Continuous Deployment

Previously, we discussed about Devnet skills, and course work briefing. We discussed Software Development and Design. I hope you had a clarity on Software Development life cycle and its development methodologies. Followed by discussion on version control systems, the importance of clean coding, overviewing object-oriented programming, Unit Testing. The last week's discussion on APIs, its designs, its use, architectural styles should have given clarity on how these technologies are used in real time environment providing better user experience. Today's lecture focuses on Application deployment.

- ✓ Previous week's Tutorial
 - Setup Lab environments, Review Linux and Python, PIP, and Python virtual environments, Version Control Systems, OOPS, Code Review and Testing, Integrate APIs
- ✓ This week's Tutorial and Independent learning
 - Continue to work with your Group.
 - **Lab 7: Create and Deploy a Sample Application**
 - **Lab 8: Building a CI/CD Pipeline using Jenkins**

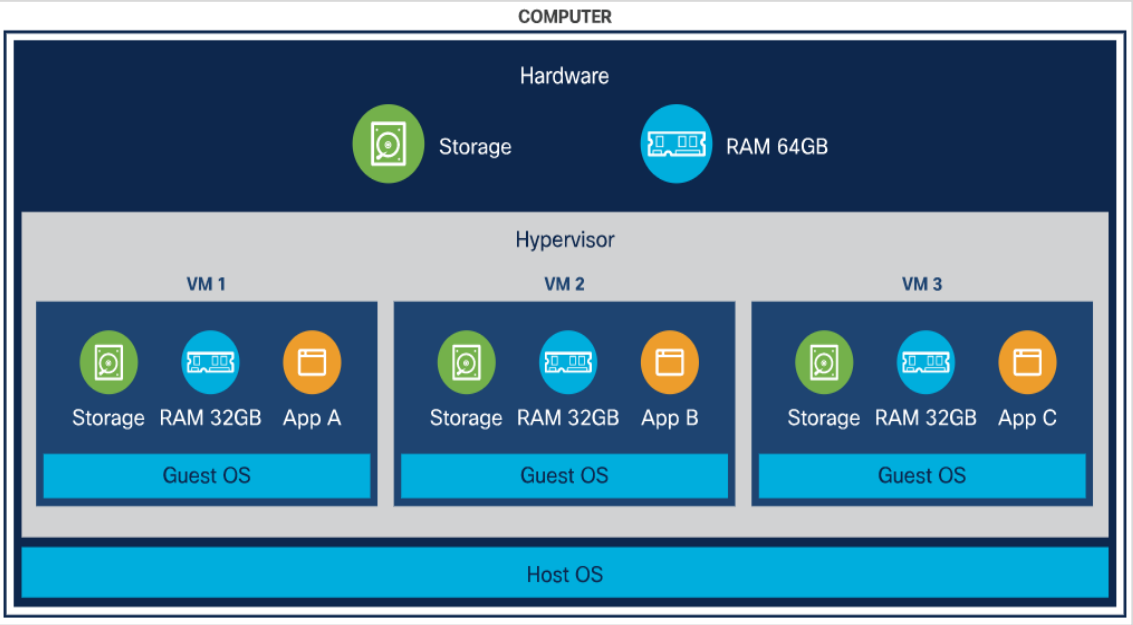
Deployment

- A piece of code, before it reaches to the user, passes through several environments focusing to improve its quality & reliability. These **environments are self-contained** and mimic the ultimate environment in which the code will live. Typically, large organizations use a four-tier structure:

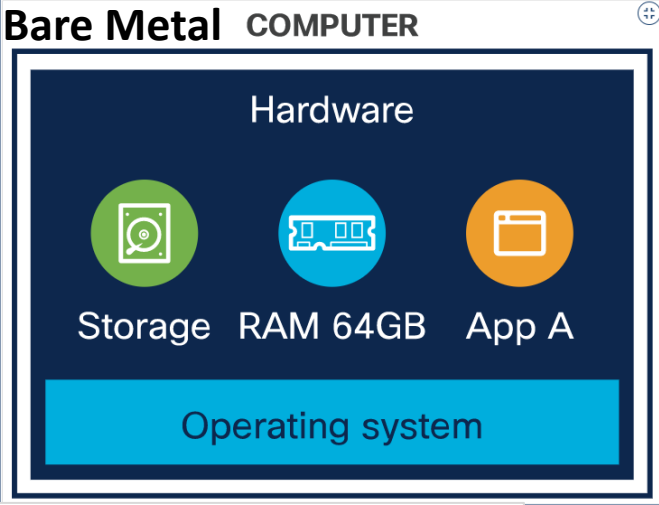
Development	Testing	Staging	Production
○ This environment is used for coding. ○ It is also used to manage fundamental aspects of the infrastructure, such as containers or cloud networking.	○ This environment is used for testing the code. ○ It should be structurally like the final production environment, on a much smaller scale. ○ It often includes automated testing tools as well as integration with a version control system.	○ This environment is used for final acceptance testing in a realistic environment. ○ After the code has been tested, it moves to the staging environment.	○ This environment is used for deploying the final code for the end user interaction. ○ It must be sized and constructed to handle expected traffic, including surges that might come seasonally or with a particular event.

Deployment Models

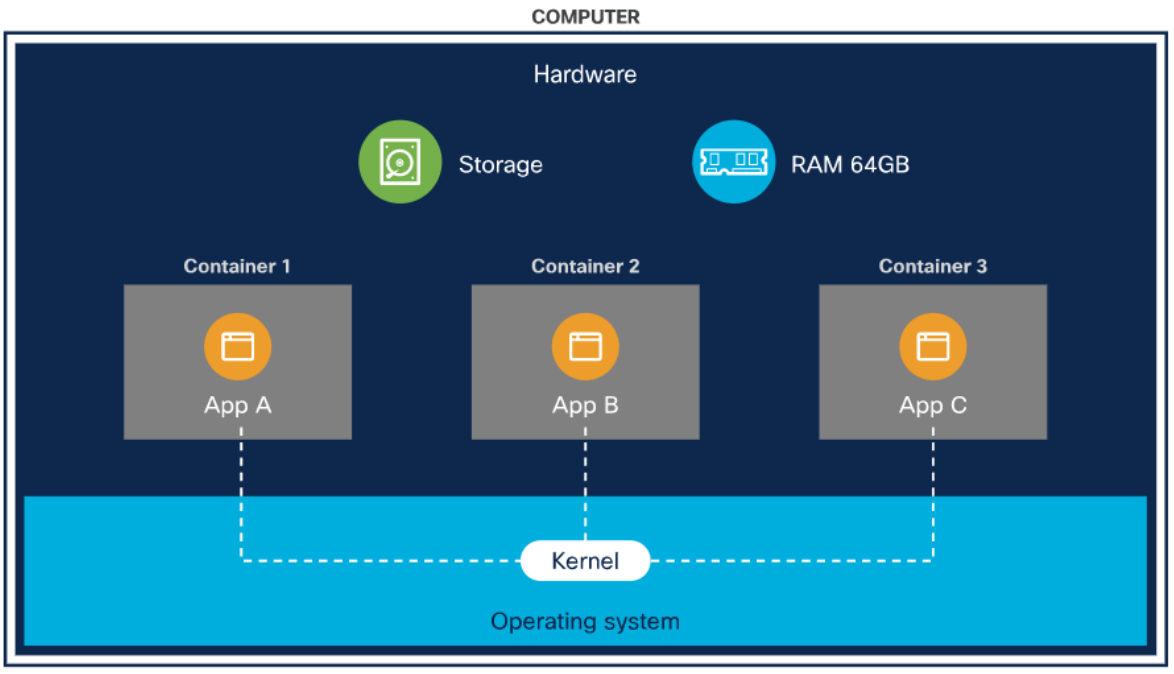
Virtual Machines (VMs)



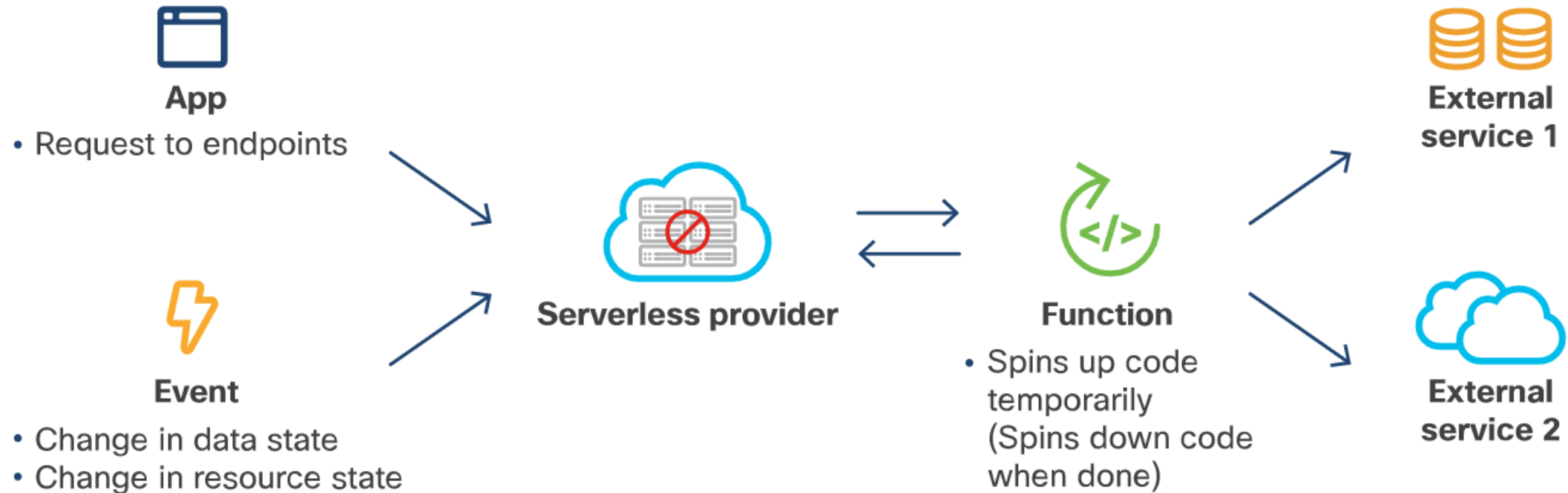
There are various deployment models that can be used to deploy a software. These include Bare Metal, Virtual Machines, Container-based Infrastructure and Serverless Computing.



Container-based infrastructure



Serverless computing takes advantage of a modern trend towards applications that are built around services.



Application makes a call to another program or workload to accomplish a particular task, to create an environment where applications are made available on an “as needed” basis.



Bare Metal



Virtual



Container

Serverless



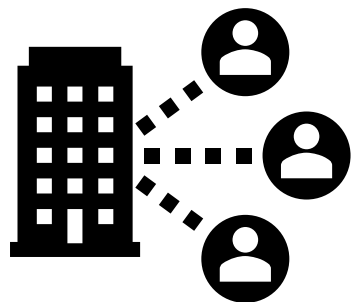
Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Trawsnewid Addysg, Trawsnewid Bywydau
Transforming Education, Transforming Lives

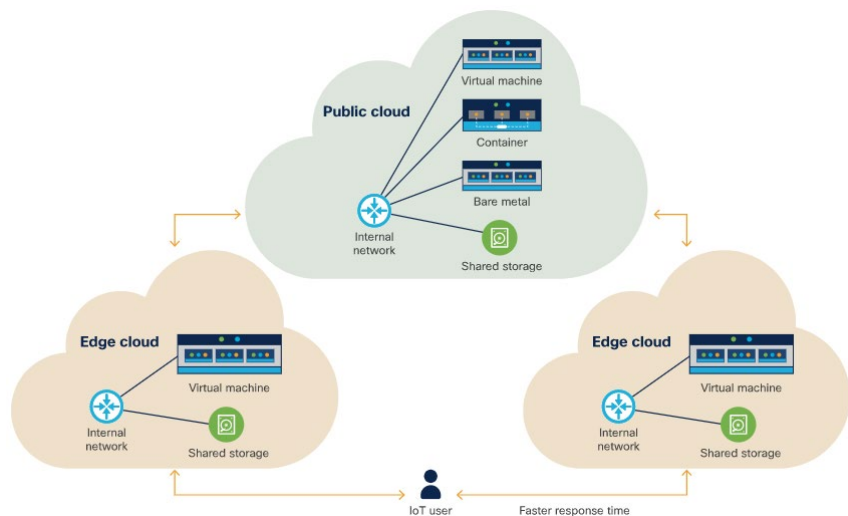
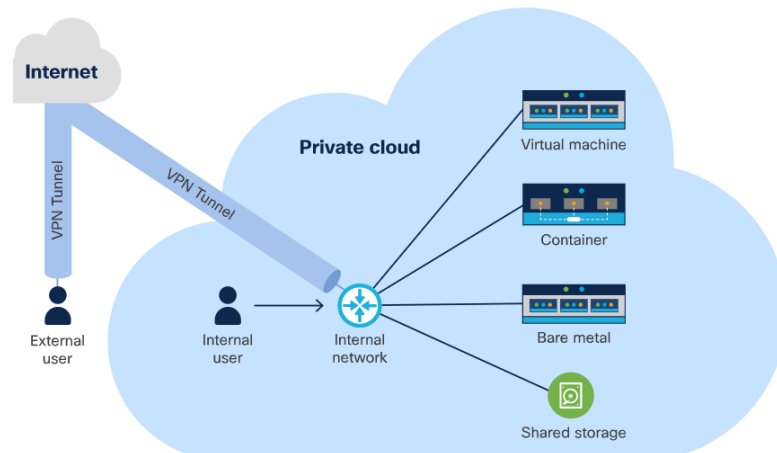
Deployment

- VM, Serverless computing, and container-based deployment are different approaches to deploying and managing applications in the cloud

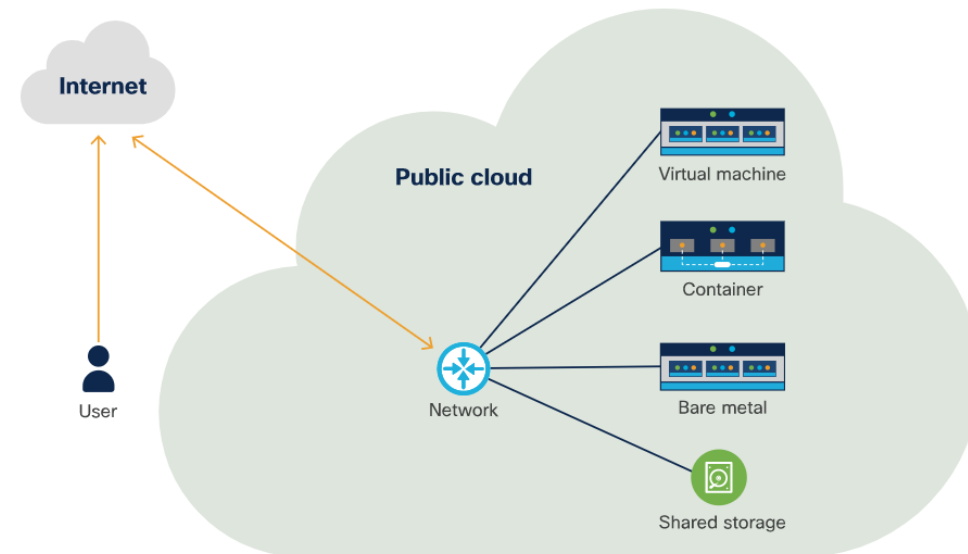
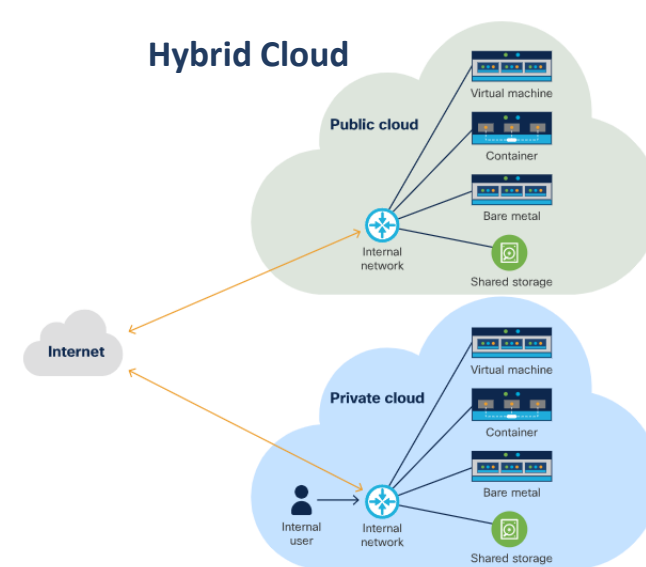
Bare Metal	VM Based	Container-based	Serverless
○ Installs and configures OS in a physical hardware. ○ Control over the hardware. ○ Human Intervention, Manual Configuration, High level maintenance	○ Runs VM on top of hypervisor, using the hardware resources ○ Flexibility and Isolation ○ Additional overhead and complexity	○ Runs applications in a container, using the OS Kernel. It requires all the required dependencies and libraries to run the application. ○ Scalability and Portability ○ More application-level configurations	○ Runs the functions in response to specific events triggered by requests. ○ On-demand resource utilization ○ Distributed across multiple instances increasing its complexity



On Premises



Infrastructure



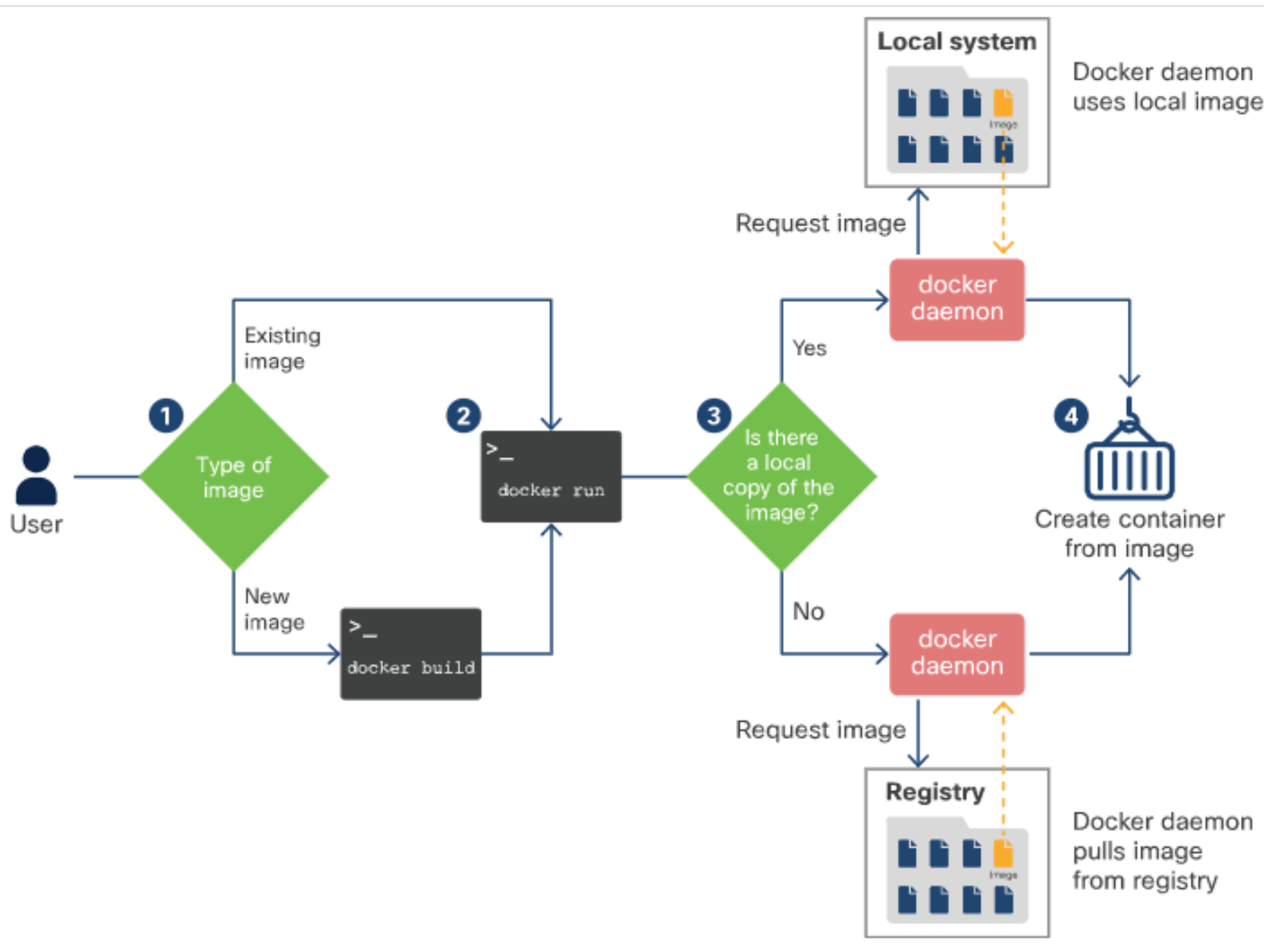
Google Drive is a public cloud storage service offered by Google

Alexa is an example of an edge cloud

Container Deployment



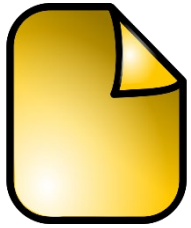
- The most **popular way to containerize an application is to deploy it as a Docker container.**
- Docker is a format that **wraps several different technologies to create containers.** These technologies are:
 - + **Namespaces** - These isolate different parts of the running container.
 - + **Control groups** - These cgroups are a standard Linux concept that enables the system to limit the resources, used by an application.
 - + **Union File Systems** - These UnionFS are file systems that are built layer by layer, combining resources.
- A Docker image is a **set of read-only files that have no state and contains source code, libraries, and other dependencies** needed to run an application.
- A **Docker container is the run-time instance of a Docker image.**
- Creating a container involves **pulling an image or a template from a repository**, then using it to create a container.



Dockerfile

is a simple text-file which is required to compile the code.

- + It defines the steps that the **docker build** command takes to create an image that can be used to create the target container.



display the client IP address

sample_app.py

Import required methods from Flask library

1

```
# A sample web application created using python.  
# Importing FLask library  
from flask import Flask  
from flask import request
```

Create an instance of the Flask class

2

```
sample = Flask(__name__)
```

Define a method to display the client IP address

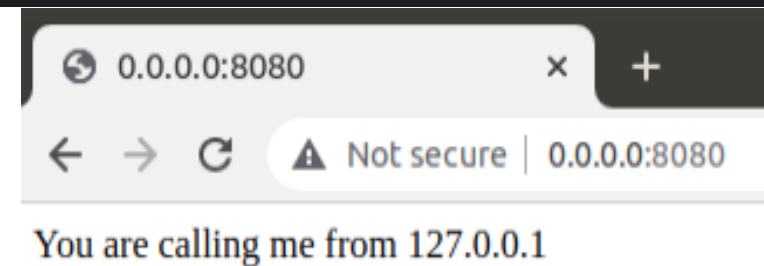
3

```
@sample.route("/")  
def main():  
    #request.remote_addr is a property in the Flask web framework  
    # It asks the IP address of the client that sent the current request.  
    # This property returns a string representing the IP address of the client machine or the last proxy that sent the request.  
  
    return "You are calling me from " + request.remote_addr + "\n"
```

Configure the app to run locally at http://0.0.0.0:8080

4

```
#Use the function run in the main  
if __name__ == "__main__":  
    sample.run(host="0.0.0.0", port=8080)
```



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Trawsnewid Addysg, Trawsnewid Bywydau
Transforming Education, Transforming Lives

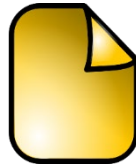
display the client IP address



sample_app.py



Index.html



style.css

```
devasc@labvm:~/labs/devnet-src/sample-app$ cat templates/index.html
<html>
<head>
  <title>My First Sample app</title>
  <link rel="stylesheet" href="/static/style.css" />
</head>
<body>
  <h1>You are calling me from {{request.remote_addr}}</h1>
  <h2> Created by - Nitheesh for accb5029 </h2>
</body>
</html>
devasc@labvm:~/labs/devnet-src/sample-app$ cat static/style.css
body {background: rgb(242, 6, 22);}
```

2

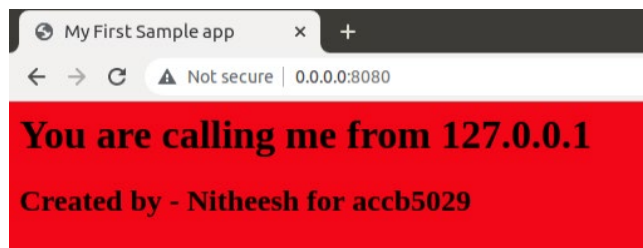
Use render template to display the client IP

```
@sample.route("/")
def main():
    #request.remote_addr is a property in the Flask web framework
    # It asks the IP address of the client that sent the current request.
    # This property returns a string representing the IP address of the client machine or the last proxy that sent the request.

    #return "You are calling me from " + request.remote_addr + "\n"
    # editing to the return function to use render_Template
    return render_template("index.html")

#Use the function run in the main
if __name__ == "__main__":
    sample.run(host="0.0.0.0", port=8080)
```

3



Import required methods from Flask library and create an instance

```
# A sample web application created using python.
# Importing FLask library
✓ from flask import Flask
from flask import request

# Importing render_template method from the flask library
# HTML file can be rendered in FLask automatically using the render_template function
from flask import render_template

sample = Flask(__name__)
```

display the client IP address



Bash File

sample_app.sh

1

```
#!/bin/bash
# command to create a directory structure with tempdir as the parent folder
mkdir tempdir
mkdir tempdir/templates
mkdir tempdir/static
```

Build the Docker image

4

```
cd tempdir
docker build -t sampleapp .
```

```
>_
docker build
```

instance of the container
(samplerunning) and the
instance docker image
(sampleapp)



Temporary Directory

Copy

sample_app.py

Templates: Index.html

Static: style.css

2

sample_app.py

Index.html

style.css

```
#Copy the website directories and sample_app.py to the temporary directory
cp sample_app.py tempdir/.
#Recursive copy the template files
cp -r templates/* tempdir/templates/.
cp -r static/* tempdir/static/.
```

```
echo "FROM python" > tempdir/Dockerfile
```

3

```
echo "RUN pip install flask" >> tempdir/Dockerfile
```

```
echo "COPY ./static /home/myapp/static/" >> tempdir/Dockerfile
echo "COPY ./templates /home/myapp/templates/" >> tempdir/Dockerfile
echo "COPY sample_app.py /home/myapp/" >> tempdir/Dockerfile
```

```
echo "EXPOSE 8080" >> tempdir/Dockerfile
```

```
echo "CMD python3 /home/myapp/sample_app.py" >> tempdir/Dockerfile
```



Docker File

Add the docker run command to the sample-app.sh file to start the container.

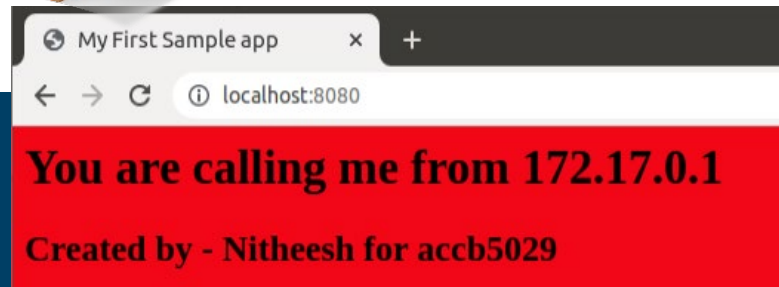
5

```
docker run -t -d -p 8080:8080 --name samplerunning sampleapp
```

6

Execute the bash script from the command line

```
labs/devnet-src/sample-app$ ./sample-app.sh
```



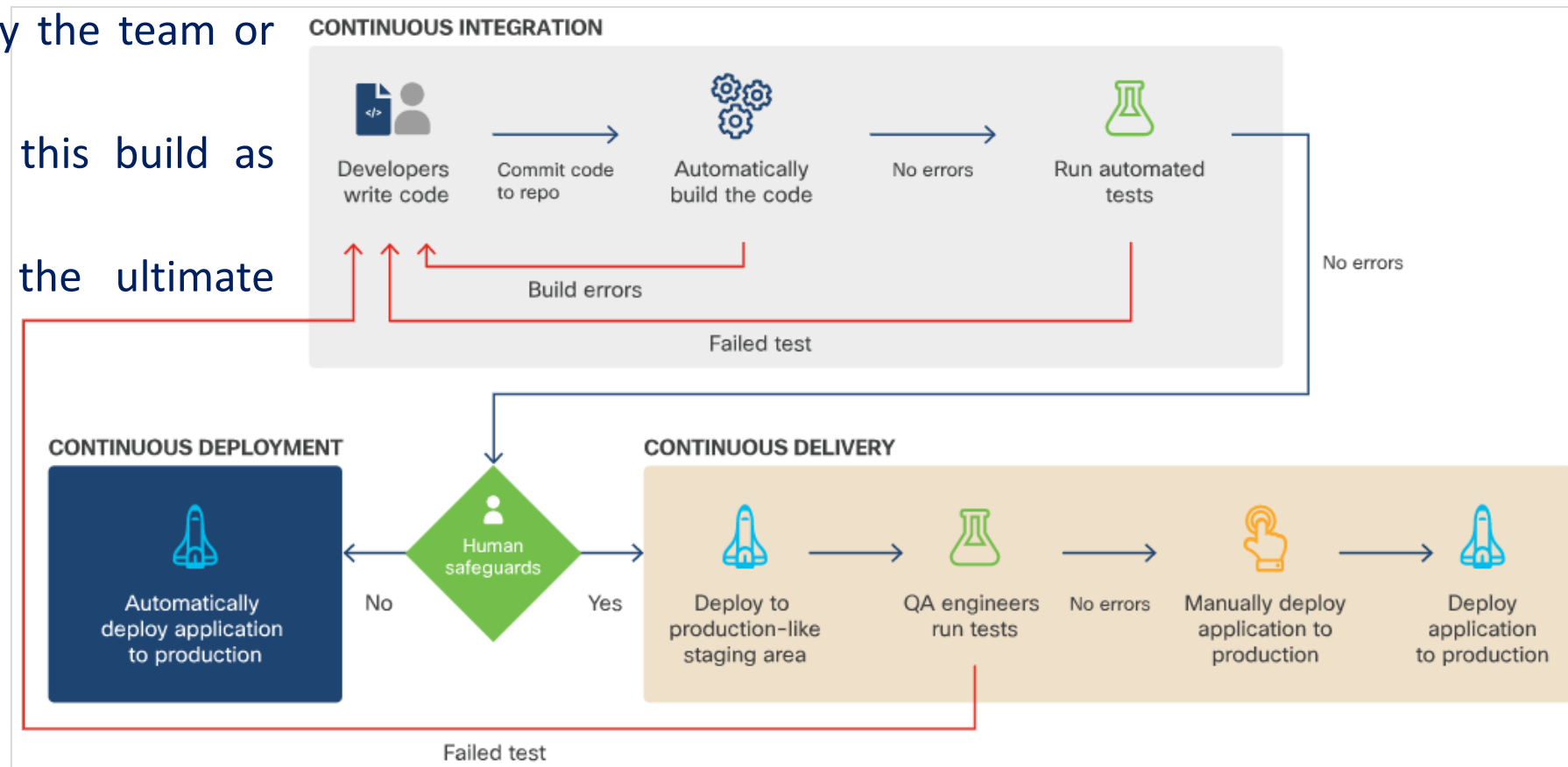
Trawsnewid Addysg, Trawsnewid Bywydau
Transforming Education, Transforming Lives

Continuous Integration/ Continuous Deployment (CI/CD)

- Continuous Integration/Continuous Deployment (CI/CD) is a philosophy for software deployment that figures prominently in the field of DevOps.
- **DevOps is about communication and making certain all members of the team are working together to ensure smooth operation.**
- Continuous Integration enables the developers on the project to ***continually merge the changes with the main branch of the existing application.***
- The Continuous Integration process provides several additional benefits:
 - + Code compilation
 - + Unit test execution
 - + Static code analysis
 - + Integration testing
 - + Packaging and versioning
 - + Publishing the version package to Docker Hub or other package repositories

Continuous Integration/ Continuous Deployment (CI/CD)

- Start with the version artifact created as part of the CI process.
- Automatically deploy the candidate version on staging.
- Run gating tests identified by the team or organization.
- If all gating tests pass, tag this build as suitable for production.
- Continuous Deployment is the ultimate expression of CI/CD.
- It is a special type of Continuous Delivery in which, every version of software that is marked as ready for production gets deployed.



Continuous Integration/ Continuous Deployment (CI/CD)

The benefits of using CI/CD for development include:

- ✓ Integration with agile methodologies
- ✓ Shorter Mean Time To Resolution (MTTR)
- ✓ Automated deployment
- ✓ Less disruptive feature releases
- ✓ Improved quality
- ✓ Improved time to market

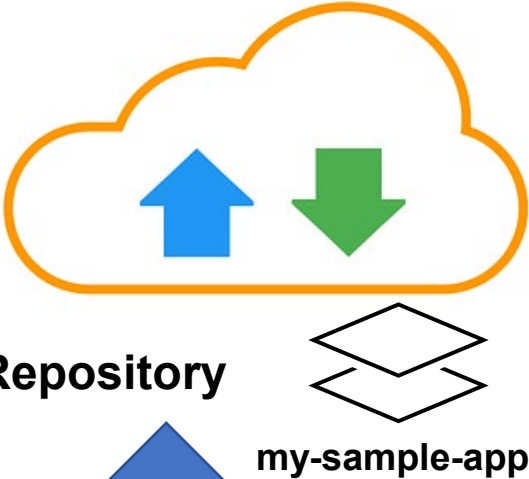
Continuous Integration/ Continuous Deployment (CI/CD)

```
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 2 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (13/13), 2.81 KiB | 410.00 KiB/s, done.
Total 13 (delta 0), reused 0 (delta 0)
To https://github.com/accb6019/my-sample-app.git
* [new branch]      master -> master
```

Push

```
[master (root-commit) 2c26341] Committing my-sample-app files after Lab 7
11 files changed, 134 insertions(+)
create mode 100644 copingtext
create mode 100644 running.txt
create mode 100755 sample-app.sh
create mode 100644 sample_app.py
create mode 100644 static/style.css
create mode 100644 tmpdir/Dockerfile
create mode 100644 tmpdir/sample_app.py
create mode 100644 tmpdir/static/style.css
create mode 100644 tmpdir/templates/index.html
create mode 100644 templates/index.html
create mode 100755 userdetails
```

Commit



1 Create a New Repository

3

Point Git Rep to GitHub

2 Initialize a directory as the Git repository

```
/labs/devnet-src/sample-app$ git remote add origin https://github.com/accb6019/my-sample-app.git
```

```
Initialized empty Git repository in /home/devasc/labs/devnet-src/sample-app/.git/
```



- sample_app.py
- Templates: Index.html
- Static: style.css

4 Stage, commit, and push the sample-app files to the GitHub repository

Stage

```
On branch master
No commits yet

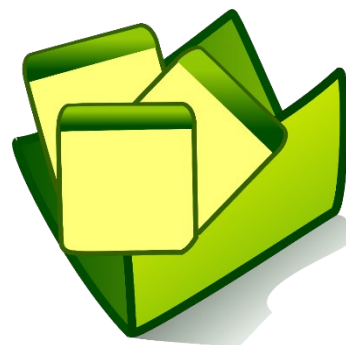
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   copingtext
    new file:   running.txt
    new file:   sample-app.sh
    new file:   sample_app.py
    new file:   static/style.css
    new file:   tmpdir/Dockerfile
    new file:   tmpdir/sample_app.py
    new file:   tmpdir/static/style.css
    new file:   tmpdir/templates/index.html
    new file:   templates/index.html
    new file:   userdetails
```

Continuous Integration/ Continuous Deployment (CI/CD)



1

Pull Jenkins Docker



sample_app.py



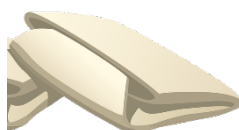
Templates: Index.html



Static: style.css

2

Configure Jenkins



Job: MySamAppBld

Source Code Management

☐ None

☒ Git

Repositories

Repository URL

<https://github.com/accb6019/my-sample-app.git>

Execute shell

Command

See [the list of available environment variables](#)

```
bash ./my-sample-app.sh
```

Build Triggers

☐ Trigger builds remotely (e.g., from scripts)

☒ Build after other projects are built

Build Steps

Execute shell

Command

```
if curl http://172.17.0.1:5050/ | grep "You are calling me from 172.17.0.1"; then
  exit 0
else
  exit 1
fi
```



my-sample-app

Pipeline FirstPipeline

Average stage times:
(Average full run time: ~19s)

#2	Mar 14	No Changes
	19:03	

Preparation	Build	Results
7s	5s	6s
650ms	11s	6s

Pipeline script

Script

```
1 node {
2   stage('Preparation') {
3     catchError(buildResult: 'SUCCESS') {
4       sh 'docker stop samplerunning'
5       sh 'docker rm samplerunning'
6     }
7   }
8   stage('Build') {
9     build 'BuildAppJob'
10  }
11  stage('Results') {
12    build 'TestAppJob'
13  }
14 }
```



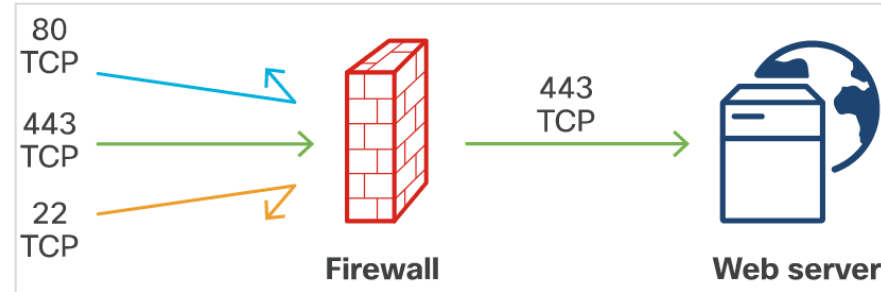
Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Trawsnewid Addysg, Trawsnewid Bywydau
Transforming Education, Transforming Lives

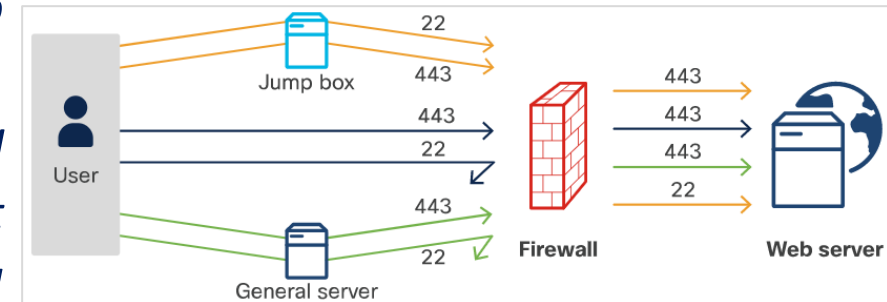
Security

- Some of the applications which need to be considered for cloud deployment are given below:

- ✓ **Firewalls**
- ✓ Load balancers
- ✓ DNS
- ✓ Reverse proxies



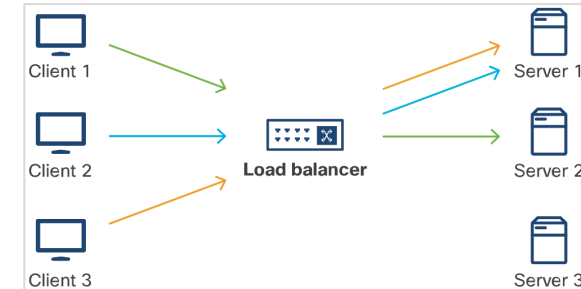
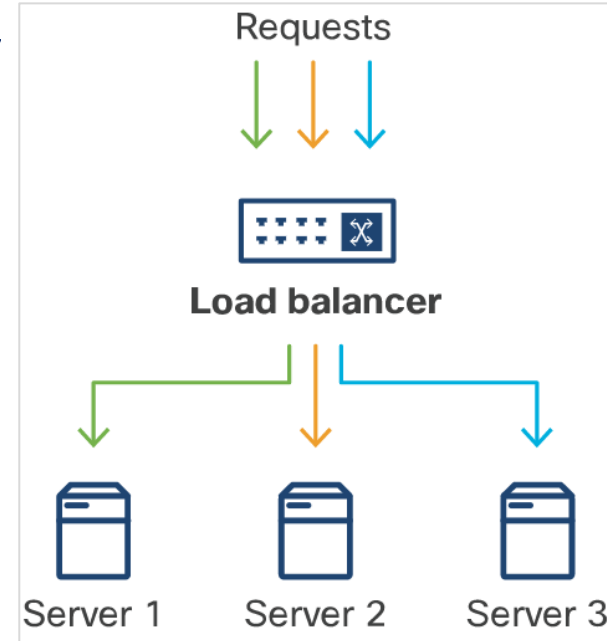
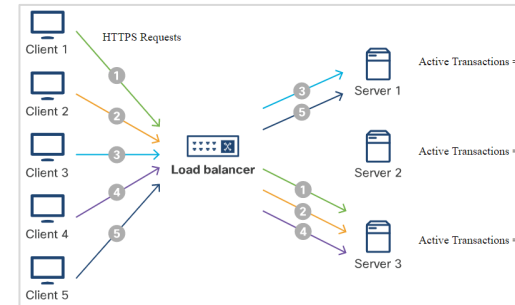
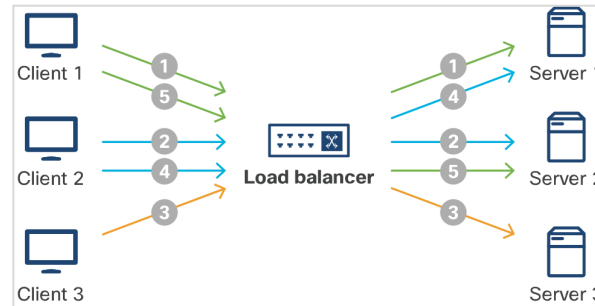
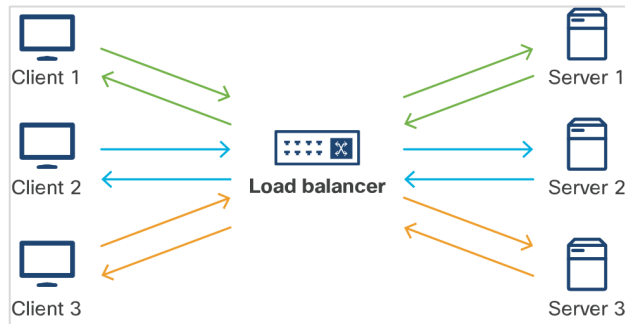
- You might sometimes set up your systems so that **logins to sensitive systems can only come from a single machine**. This is called a “jump box”.
- A jump box can be used to provide additional access while still providing an additional layer of security. It sets up the systems so that logins can only come from a single machine, and everyone must log into that server first, then log into the target machine from there.



Security

○ Some of the applications which need to be considered for cloud deployment are given below:

- ✓ Firewalls
 - ✓ **Load balancers**
 - ✓ DNS
 - ✓ Reverse proxies
- *A load balancer takes requests and balances them by spreading them out among multiple servers.*
 - *A load balancer parcels out requests to different servers.*
 - *Load balancers make their decisions on which servers should get a particular request in a few different ways.*



Security

Securing the Data: Best practices for storing encrypted data

- Data breaches occur **when data is stored but not protected**. When it comes to protecting data at rest, there are a few things to consider.
 - ✓ **Encrypting data:** Data encryption ensures that when an unauthorized access is gained into the system, the data is not visible in its actual form. There are two methods for encrypting data:

One-way encryption	Two-way encryption
One way encryption is simpler: create an encrypted value without necessarily using a specific key, but you cannot unencrypt it. <i>You would use that for information you do not need to retrieve, just need to compare, such as passwords.</i>	In Two-way encryption: Encrypt the data using a key, and then you can use that key (or a variation on it) to decrypt the data to get it back in plaintext. <i>You would use this for information you would need to access in its original form, such as medical records or social security numbers.</i>

Security

Software vulnerabilities

- Keeping up with the latest vulnerabilities and patching them as appropriate.
- Storing too much data: Unless the data is needed for an essential function, don't store it.
- Storing data in the cloud: Remember that when storing data in the cloud, it is stored in someone else's computer. Make sure that your cloud data is encrypted or otherwise protected.
- Roaming devices: Apps are increasingly on devices that even more portable than laptops, such as tablets and especially cell phones. They are simply easier to lose. Be sure you are not leaving your data vulnerable by encrypting it whenever possible.
- Data is also vulnerable when it is being transmitted. The following can be used to prevent data vulnerability problems:
 - ✓ **SSH** - SSH provides authentication and encryption of messages between the source and target machines, making it difficult or impossible to snoop on the users' actions.
 - ✓ **TLS** - TLS provides message authentication and stronger ciphers than SSL.
 - ✓ **VPN** - A VPN keeps all application-related traffic inside the network, which acts as a proxy and encrypts all traffic to and from the user.

Security

Authentication

- Incorporating other authentication factors confuses the hackers who may have cracked the password. Some of these factors are as follows:
- **Single-factor authentication (SFA)**
- **Two-factor authentication (2FA)**
 - Two-factor authentication uses the same password/username combination, but with the addition of being asked to verify the identity of the persons by using something owned by them only such as a mobile device.
- **Multi-factor authentication (MFA)**
 - Multi-factor authentication (MFA) is a method of computer access control in which a user is only granted access after successfully presenting several separate pieces of evidence to an authentication mechanism.
 - At least two of the mentioned categories are required for MFA: knowledge; possession, and inheritance.
 - 2FA is just a type of MFA where you only need two pieces of evidence, two “factors”.

Security

Six key principles of human influence

- **Reciprocity** – Our social norms mean that we tend to return a favor when asked.
- **Commitment and consistency** – When people commit, whether in person, in writing, or on a web site, they are more likely to honor that commitment in order to preserve their self-image.
- **Social proof** – When people see someone else doing something, such as looking up, others will stop to do the same.
- **Authority** – This authority principle means that attackers who seem to be authoritative or representing an authority figure are more likely to gain access.
- **Liking** – Likable people are able to persuade others more effectively. People are easily persuaded by familiar people whom they like.
- **Scarcity** – When people believe that something is limited in amount, people will act positively and quickly to pick up the desired item.
- There are **four social engineering vectors**, or lines of attack, that can take advantage of these influence principles. **Phishing, Vishing, Smishing, Impersonation.**

Security

What is OWASP?

- The Open Web Application Security Project (OWASP) is focused on providing education, tools, and other resources to help developers avoid security problems in web-based applications. Resources provided by OWASP include:

Tools	Code Projects	Documentation Projects
• OWASP Zed Attack Proxy (ZAP) • Dependency Check • OWASP DefectDojo	• OWASP ModSecurity Core Rule Set (CRS) • OWASP CSRFGuard	• OWASP Application Security Verification Standard • OWASP Top Ten • OWASP Cheat Sheet Series

- OWASP recommends never displaying untrusted content in the following locations:
 - ✓ Inside script tags
 - ✓ Inside comments
 - ✓ As part of attribute names
 - ✓ As part of tag names
 - ✓ In CSS (within style tags)

Security

OWASP TOP TEN

- **Injection:** This includes all sorts of injection attacks that can be prevented by using parameterized APIs, escaping user input, and by using LIMIT clauses.
- **Broken Authentication:** This relates to multiple problems with user credentials. These attacks can be prevented by avoiding default passwords, using multi-factor authentication, and by using techniques like lengthening waiting periods after failed logins.
- **Sensitive Data Exposure:** This refers to scenarios when attackers steal sensitive information. Such scenarios can be prevented by storing as little personal information as possible, and by using encryption.
- **XML External Entities (XXE):** These are attacks made possible an XML feature that permits incorporating external information using entities and can be prevented by disabling XML Entity and DTD processing, or by using JSON format.
- **Broken Access Control:** This refers to the need to ensure that an application that enables users to circumvent existing authentication requirements should not be built and can be avoided by protecting all resources and functions on the server side.

Security

OWASP TOP TEN

- **Security Misconfiguration:** This refers to the need to ensure that the system itself is properly configured. Prevention of these types of problems requires careful, consistent hardening of systems and applications.
- **Cross-Site Scripting (XSS):** This refers to the ability for an attacker to use the dynamic functions of a site to inject malicious content into the page. These attacks can be prevented by carefully considering where to include the untrusted content as well as sanitizing any untrusted content.
- **Insecure Deserialization:** This describes issues that can occur if attackers can access, and potentially change, serialized versions of data and objects. To prevent such issues, do not accept serialized objects from untrusted sources.
- **Using Components with Known Vulnerabilities:** Most of the core functions are probably been written and included in an existing software package, and it is probably open source. Many of the packages that are available also include publicly available exploits. To fix this, ensure that they are using only necessary features and secure packages.
- **Insufficient Logging and Monitoring:** It is important to ensure that the logs are in a common format so that they can be easily consumed by reporting tools, and that they are auditable to detect tampering.

Feedback/ Any Questions?

Module Expectations and Daily Feedback - ACCB5029

Tue, Oct 07, 2025

<https://vevox.app/#/m/136436531>

Session ID: 136-436-531

