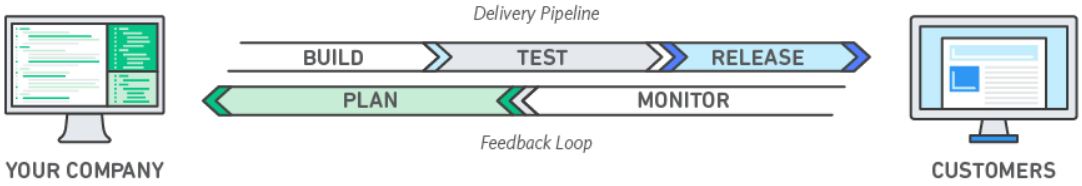
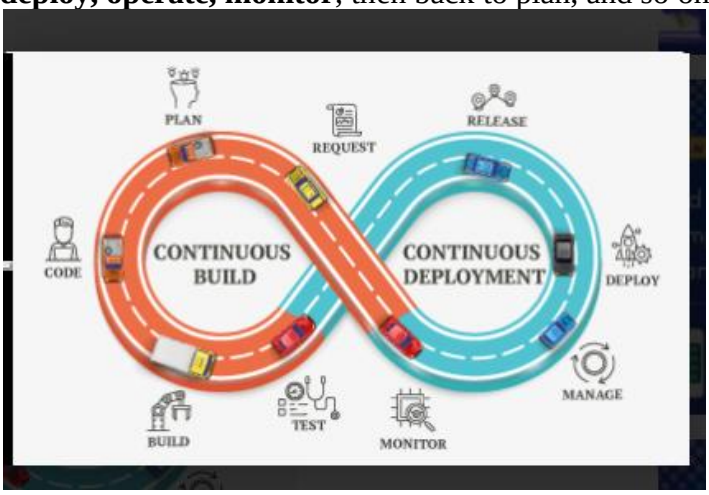
Week 3

Overview of DevOps

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.



DevOps is visualized as an infinite loop comprising the steps: **plan, code, build, test, release, deploy, operate, monitor**, then back to plan, and so on.



DevOps is an evolving philosophy and framework that encourages faster, better application development and faster release of new or revised software features or products to customers.

The practice of DevOps encourages smoother, continuous communication, collaboration, integration, visibility, and transparency between application development teams (Dev) and their IT operations team (Ops) counterparts.

This closer relationship between "Dev" and "Ops" permeates every phase of the DevOps lifecycle: from initial software planning to code, build, test, and release phases and on to deployment, operations, and ongoing monitoring.

Configuration management

Configuration management is all about trying to ensure that the files and software you are expecting to be on a machine are present, configured correctly, and working as intended.
Configuration management is important in DevOps because it helps you automate and allows an organization to increase ability to move quickly and easily otherwise it will become too long tasks. Moreover, configuration management supports DevOps holistically, and it's widely agreed upon that configuration management is essential to DevOps, as opposed to being just another component of the process.

Continuous integration

- Continuous integration is a process in devops where changes are merged into a central repository after which the code is automated and tested. The continuous integration process

is a practice in software engineering used to merge developers' working copies several times a day into a shared mainline.

- It refers to the process of automating the integration of code changes coming from several sources. The process comprises several automation tools that emphasize on the code's correctness before Integration.
- Continuous Integration (CI) is a software development practice where developers regularly merge their code changes into a central code repository, after which automated builds and tests are run. CI helps find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

Automated testing

Test automation helps development teams build, test, and ship faster and more reliably.

Test automation is the practice of automatically reviewing and validating a software product, such as a web application, to make sure it meets predefined quality standards for code style, functionality (business logic), and user experience.

Testing practices typically involve the following stages:

- **Unit testing**: validates individual units of code, such as a function, so it works as expected
- **Integration testing**: ensures several pieces of code can work together without unintended consequences
- **End-to-end testing**: validates that the application meets the user's expectations
- **Exploratory testing**: takes an unstructured approach to reviewing numerous areas of an application from the user perspective, to uncover functional or visual issues

Infrastructure as code

This practice can be used during various DevOps phases to automate the provisioning of infrastructure required for a software release. Developers add infrastructure "code" from within their existing development tools.

For example, developers might create a storage volume on demand from Docker, Kubernetes, or OpenShift. This practice also allows operations teams to monitor environment configurations, track changes, and simplify the rollback of configurations.

Continuous delivery

- Continuous delivery is a DevOps software development practice where "code changes are automatically built, tested, and prepared for a release to production. Continuous delivery expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage."
- **Continuous delivery** is a software development practice where code changes are automatically prepared for a release to production. A pillar of modern application

development, continuous delivery expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage

Continuous deployment (CD)

Similar to continuous delivery, this practice automates the release of new or changed code into production. A company doing continuous deployment might release code or feature changes several times per day. The use of container technologies, such as Docker and Kubernetes, can enable continuous deployment by helping to maintain consistency of the code across different deployment platforms and environments.

Continuous monitoring

This practice involves ongoing monitoring of both the code in operation and the underlying infrastructure that supports it. A feedback loop that reports on bugs or issues then makes its way back to development

Git hub

Git is a free and open-source version control system, originally created by Linus Torvalds in 2005.

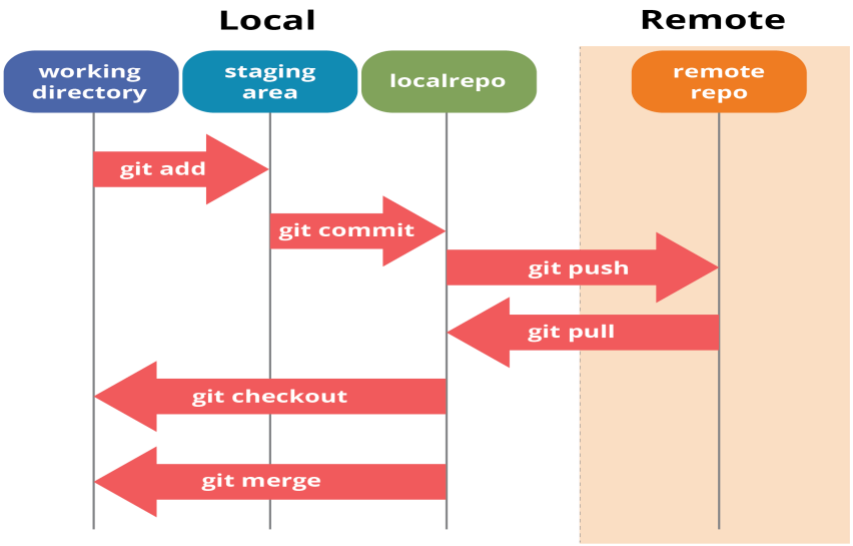
Git is distributed: every developer has the full history of their code repository locally. This makes the initial clone of the repository slower, but subsequent operations such as commit, blame, diff, merge, and log dramatically faster.

Git also has excellent support for branching, merging, and rewriting repository history, which has led to many innovative and powerful workflows and tools. Pull requests are one such popular tool that allows teams to collaborate on Git branches and efficiently review each other's code. Git is the most widely used version control system in the world today and is considered the modern standard for software development.

Basic local git operations

Basic overview of how Git works:

1. Create a "repository" (project) with a git hosting tool (like Bitbucket)
2. Copy (or clone) the repository to your local machine
3. Add a file to your local repo and "commit" (save) the changes
4. "Push" your changes to your main branch
5. Make a change to your file with a git hosting tool and commit
6. "Pull" the changes to your local machine
7. Create a "branch" (version), make a change, commit the change
8. Open a "pull request" (propose changes to the main branch)
9. "Merge" your branch to the main branch



Creating a repository

Create a "repository" (project) with a git hosting tool (like Bitbucket)

Cloning a repository - git clone
In order to work with code, you must have a local copy of it. You will find the path to clone at the top of the fork.


```
git clone git@git.yale.edu:drupal/d7.git
```

Making and recording changes - git add
Git has a staging area that modified files must be added to before they can be committed. This allows for easily breaking changes up into separate commits.

Example: *Add an individual file:*

```
git add filename
```

Add all modified and deleted files:

```
git add -A
```

Staging and committing changes - git commit
Records a snapshot of your changes. Always include a comment with what was modified.

Example:

```
git commit -m"Added CSS to style the home page."
```

git push

Your commits aren't pushed immediately to the remote repository when committing. This pushes your changes to the remote repository.

git pull –rebase

If changes have been pushed to your fork by other users, you will need to pull them in before you can push. This pulls those changes in and applies your changes on top of them.

Viewing the history of all the changes undoing changes - git log

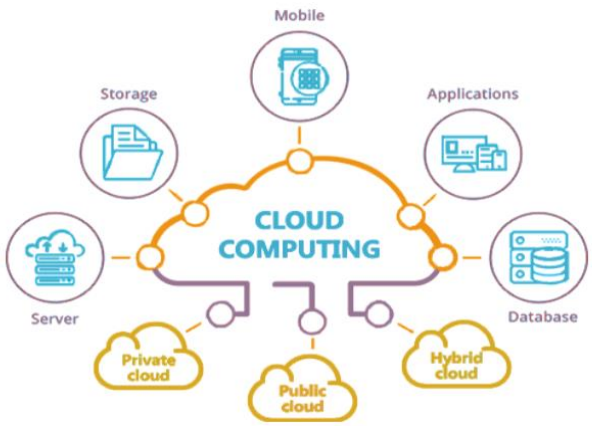
The git log command shows the order of the commit history for a repository.

The command helps in understanding the state of the current branch by showing the commits that lead to this state.

CLOUD BASICS

The cloud" refers to servers that are accessed over the Internet, and the software and databases that run on those servers. Cloud servers are located in data centers all over the world.

- Cloud Infrastructure Overview - Cloud computing infrastructure is the collection of hardware and software elements needed to enable cloud computing. It includes computing power, networking, and storage, as well as an interface for users to access their virtualized resources.



- Cloud Computing Architecture And its Components -

cloud computing technology is used by both small and large organizations to store the information in cloud and access it from anywhere at anytime using the internet connection.

Cloud computing architecture is a combination of service-oriented architecture and event-driven architecture.

Cloud computing architecture is divided into the following two parts -

- Front End
- Back End

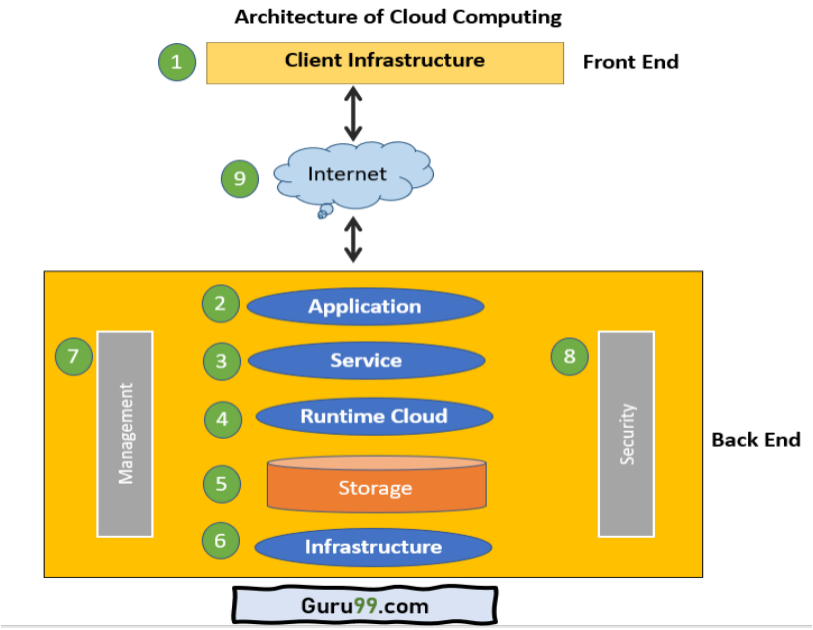
Front End

The front end is used by the client. It contains client-side interfaces and applications that are required to access the cloud computing platforms. The front end includes web servers (including Chrome, Firefox, internet explorer, etc.), thin & fat clients, tablets, and mobile devices.

Back End

The back end is used by the service provider. It manages all the resources that are required to provide cloud computing services. It includes a huge amount of data storage, security mechanism, virtual machines, deploying models, servers, traffic control mechanisms, etc.

The below diagram shows the architecture of cloud computing -



Components of Cloud Computing Architecture

There are the following components of cloud computing architecture -

1. Client Infrastructure

Client Infrastructure is a Front end component. It provides GUI (Graphical User Interface) to interact with the cloud.

2. Application

The application may be any software or platform that a client wants to access.

3. Service

A Cloud Services manages that which type of service you access according to the client's requirement.

Cloud computing offers the following three type of services:

i. Software as a Service (SaaS)

ii. Platform as a Service (PaaS)

iii. Infrastructure as a Service (IaaS)

4. Runtime Cloud

Runtime Cloud provides the **execution and runtime environment** to the virtual machines.

5. Storage

Storage is one of the most important components of cloud computing. It provides a huge amount of storage capacity in the cloud to store and manage data.

6. Infrastructure

It provides services on the **host level**, **application level**, and **network level**. Cloud infrastructure includes hardware and software components such as servers, storage, network devices, virtualization software, and other storage resources that are needed to support the cloud computing model.

7. Management

Management is used to manage components such as application, service, runtime cloud, storage, infrastructure, and other security issues in the backend and establish coordination between them.

8. Security

Security is an in-built back end component of cloud computing. It implements a security mechanism in the back end.

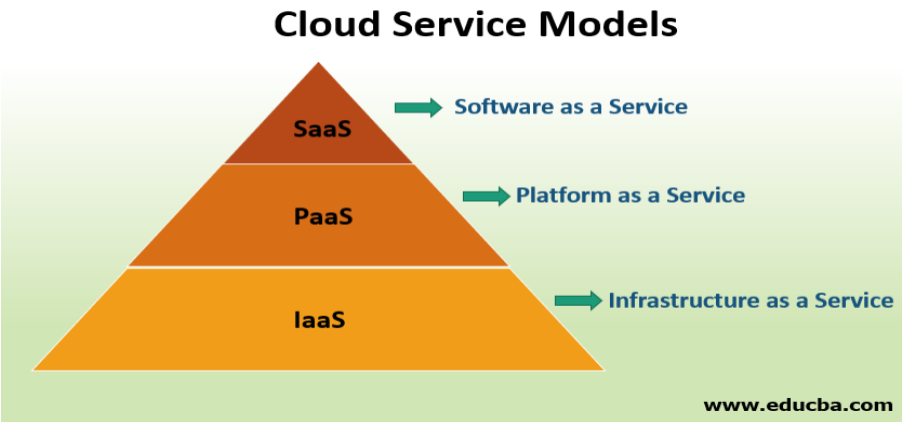
9. Internet

The Internet is medium through which front end and back end can interact and communicate with each other.

Cloud Service Models

There are the following three types of cloud service models -

1. Infrastructure as a Service (Iaas)
2. Platform as a Service (Paas)
3. Software as a Service (Saas)



Infrastructure as a Service (IaaS)

IaaS is also known as **Hardware as a Service (HaaS)**. It is a computing infrastructure managed over the internet. The main advantage of using IaaS is that it helps users to avoid the cost and complexity of purchasing and managing the physical servers.

Characteristics of IaaS

There are the following characteristics of IaaS -

- Resources are available as a service
- Services are highly scalable
- Dynamic and flexible
- GUI and API-based access
- Automated administrative tasks

Example: DigitalOcean, Linode, Amazon Web Services (AWS), Microsoft Azure, Google Compute Engine (GCE), Rackspace, and Cisco Metacloud.

Platform as a Service (PaaS)

PaaS cloud computing platform is created for the programmer to develop, test, run, and manage the applications.

Characteristics of PaaS

There are the following characteristics of PaaS -

- Accessible to various users via the same development application.
- Integrates with web services and databases.
- Builds on virtualization technology, so resources can easily be scaled up or down as per the organization's need.
- Support multiple languages and frameworks.
- Provides an ability to "**Auto-scale**".

Example: AWS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App Engine, Apache Stratos, Magento Commerce Cloud, and OpenShift.

Software as a Service (SaaS)

SaaS is also known as "**on-demand software**". It is a software in which the applications are hosted by a cloud service provider. Users can access these applications with the help of internet connection and web browser.

Characteristics of SaaS

There are the following characteristics of SaaS -

- Managed from a central location
- Hosted on a remote server
- Accessible over the internet
- Users are not responsible for hardware and software updates. Updates are applied automatically.
- The services are purchased on the pay-as-per-use basis

Example: BigCommerce, Google Apps, Salesforce, Dropbox, ZenDesk, Cisco WebEx, ZenDesk, Slack, and GoToMeeting.

What Is A Cloud Deployment Model?

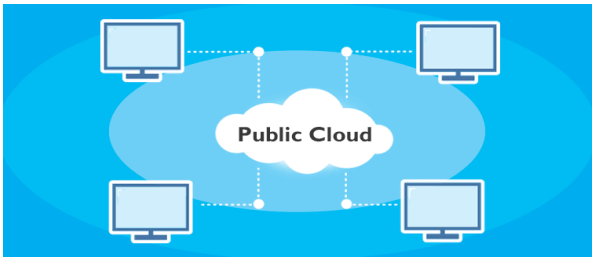
It works as your virtual computing environment with a choice of deployment model depending on how much data you want to store and who has access to the Infrastructure.

Different Types Of Cloud Computing Deployment Models

Most cloud hubs have tens of thousands of servers and storage devices to enable fast loading. It is often possible to choose a geographic area to put the data "closer" to users. Thus, deployment models for cloud computing are categorized based on their location. To know which model would best fit the requirements of your organization, let us first learn about the various types.

Public Cloud

The name says it all. It is accessible to the public. Public deployment models in the cloud are perfect for organizations with growing and fluctuating demands. It also makes a great choice for companies with low-security concerns. Thus, you pay a cloud service provider for networking services, compute virtualization & storage available on the public internet. It is also a great delivery model for the teams with development and testing. Its configuration and deployment are quick and easy, making it an ideal choice for test environments.



Benefits of Public Cloud

- Minimal Investment - As a pay-per-use service, there is no large upfront cost and is ideal for businesses who need quick access to resources
- No Hardware Setup - The cloud service providers fully fund the entire Infrastructure
- No Infrastructure Management - This does not require an in-house team to utilize the public cloud.

Private Cloud

Companies that look for cost efficiency and greater control over data & resources will find the private cloud a more suitable choice.

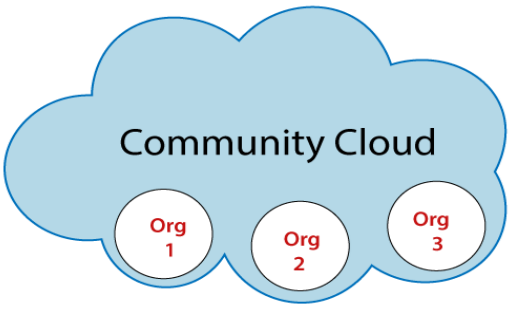
It means that it will be integrated with your data center and managed by your IT team. Alternatively, you can also choose to host it externally. The private cloud offers bigger opportunities that help meet specific organizations' requirements when it comes to customization. It's also a wise choice for mission-critical processes that may have frequently changing requirements.

Benefits of Private Cloud

- Data Privacy - It is ideal for storing corporate data where only authorized personnel gets access
- Security - Segmentation of resources within the same Infrastructure can help with better access and higher levels of security.
- Supports Legacy Systems - This model supports legacy systems that cannot access the public cloud.

Community Cloud

The community cloud operates in a way that is similar to the public cloud. There's just one difference - it allows access to only a specific set of users who share common objectives and use cases. This type of deployment model of cloud computing is managed and hosted internally or by a third-party vendor. However, you can also choose a combination of all three.



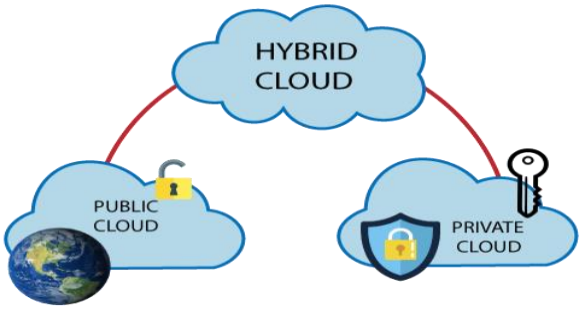
Benefits of Community Cloud

- Smaller Investment - A community cloud is much cheaper than the private & public cloud and provides great performance
- Setup Benefits - The protocols and configuration of a community cloud must align with industry standards, allowing customers to work much more efficiently.

Hybrid Cloud

As the name suggests, a hybrid cloud is a combination of two or more cloud architectures. While each model in the hybrid cloud functions differently, it is all part of the same architecture. Further, as part of this deployment of the cloud computing model, the internal or external providers can offer resources.

Let's understand the hybrid model better. A company with critical data will prefer storing on a private cloud, while less sensitive data can be stored on a public cloud. The hybrid cloud is also frequently used for 'cloud bursting'. It means, supposes an organization runs an application on-premises, but due to heavy load, it can burst into the public cloud.



Benefits of Hybrid Cloud

- Cost-Effectiveness - The overall cost of a hybrid solution decreases since it majorly uses the public cloud to store data.
- Security - Since data is properly segmented, the chances of data theft from attackers are significantly reduced.
- Flexibility - With higher levels of flexibility, businesses can create custom solutions that fit their exact requirements

Important Factors to Consider	Public	Private	Community	Hybrid
Setup and ease of use	Easy	Requires professional IT Team	Requires professional IT Team	Requires professional IT Team
Data Security and Privacy	Low	High	Very High	High

Scalability and flexibility	High	High	Fixed requirements	High
Cost-Effectiveness	Most affordable	Most expensive	Cost is distributed among members	Cheaper than private but more expensive than public
Reliability	Low	High	Higher	High

What is the concept behind the Virtualization?

Creation of a virtual machine over existing operating system and hardware is known as Hardware Virtualization. A Virtual machine provides an environment that is logically separated from the underlying hardware.

The machine on which the virtual machine is going to create is known as **Host Machine** and that virtual machine is referred as a **Guest Machine**

Types of Virtualization:

1. Hardware Virtualization.
2. Operating system Virtualization.
3. Server Virtualization.
4. Storage Virtualization.

1) Hardware Virtualization:

When the virtual machine software or virtual machine manager (VMM) is directly installed on the hardware system is known as hardware virtualization.

The main job of hypervisor is to control and monitoring the processor, memory and other hardware resources.

After virtualization of hardware system we can install different operating system on it and run different applications on those OS.

Usage:

Hardware virtualization is mainly done for the server platforms, because controlling virtual machines is much easier than controlling a physical server.

2) Operating System Virtualization:

When the virtual machine software or virtual machine manager (VMM) is installed on the Host operating system instead of directly on the hardware system is known as operating system virtualization.

Usage:

Operating System Virtualization is mainly used for testing the applications on different platforms of OS.

3) Server Virtualization:

When the virtual machine software or virtual machine manager (VMM) is directly installed on the Server system is known as server virtualization.

Usage:

Server virtualization is done because a single physical server can be divided into multiple servers on the demand basis and for balancing the load.

4) Storage Virtualization:

Storage virtualization is the process of grouping the physical storage from multiple network storage devices so that it looks like a single storage device.

Storage virtualization is also implemented by using software applications.

Usage:

Storage virtualization is mainly done for back-up and recovery purposes.

What is a cloud-native application?

A cloud-native application is a program that is designed for a cloud computing architecture. These applications are run and hosted in the cloud and are designed to capitalize on the inherent characteristics of a cloud computing software delivery model. A native app is software that is developed for use on a specific platform or device.

Cloud-native applications use a micro service architecture. This architecture efficiently allocates resources to each service that the application uses, making the application flexible and adaptable to a cloud architecture.

Proponents of DevOps use cloud-native applications for their ability to promote business agility. They are designed, built and delivered differently from traditional cloud-based monolithic applications. Cloud-native applications feature shorter application lifecycles and are highly resilient, manageable and observable.

Features of a cloud-native application

The micro services that are part of the cloud-native app architecture are packaged in containers that connect and communicate via APIs. Orchestration tools are used to manage all of these components.

Here are some of the key capabilities of these applications:

- **Microservices-based.** Microservices break down an application into a series of independent services, or modules. Each service references its own data and supports a specific business goal. These modules communicate with one another via application program interfaces (APIs).
- **Container-based.** Containers are a type of software that logically isolates the application enabling it to run independent of physical resources. Containers keep microservices from interfering with one another. They keep applications from consuming all the host's shared resources. They also enable multiple instances of the same service.
- **API-based.** APIs connect microservices and containers while providing simplified maintenance and security. They enable microservices to communicate, acting as the glue between the loosely coupled services.
- **Dynamically orchestrated.** Container orchestration tools are used to manage container lifecycles, which can become complex. Container orchestration tools handle resource management, load balancing, scheduling of restarts after an internal failure and provisioning and deploying containers onto server cluster nodes.