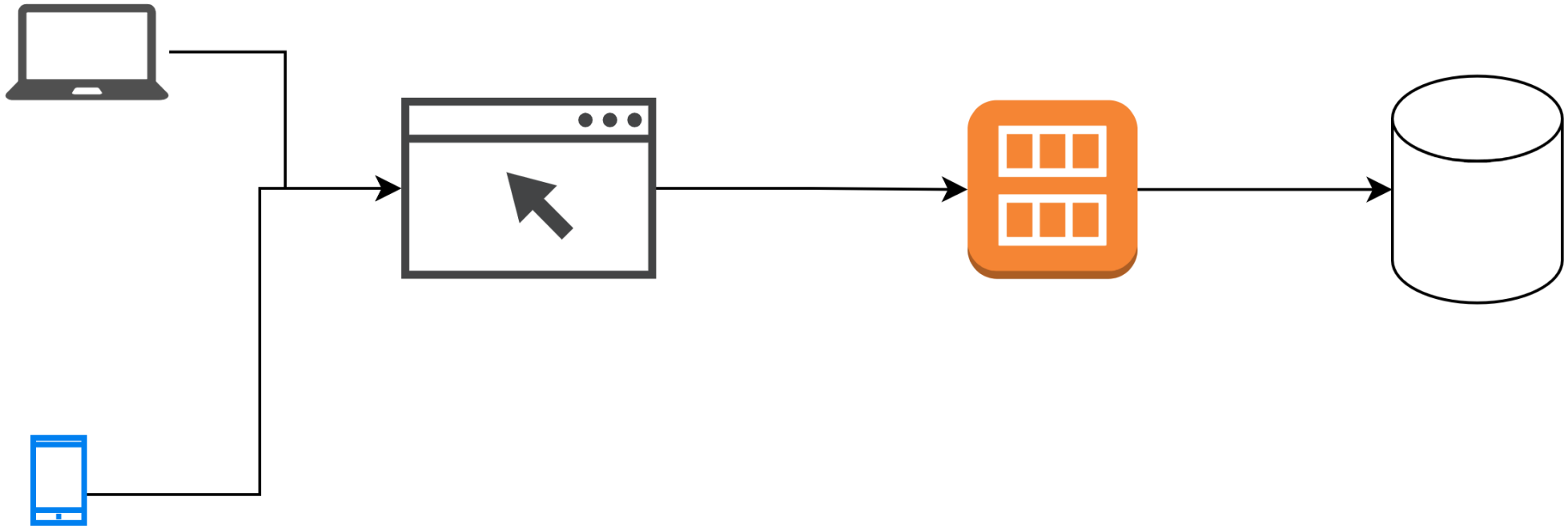
DevOps

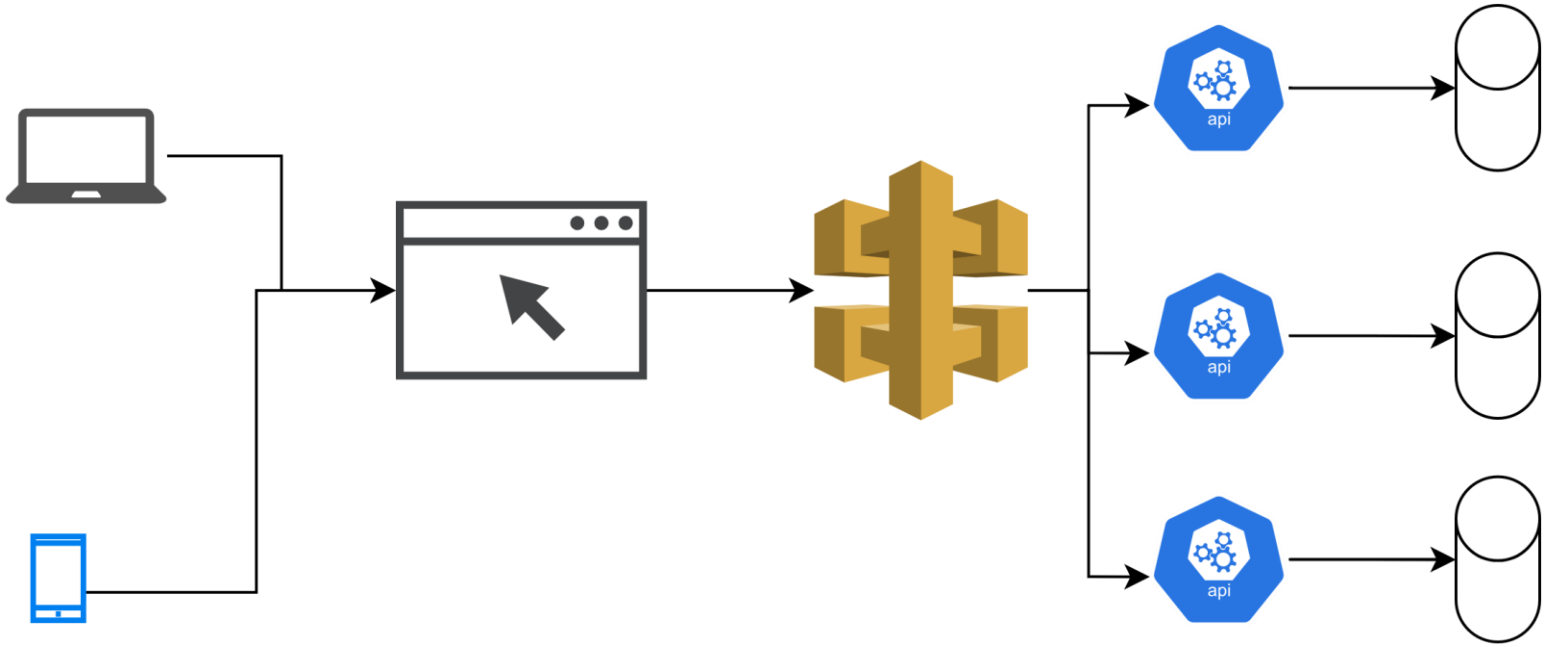
Week 13

Murtaza Munawar Fazal

Monolithic Services



Microservices



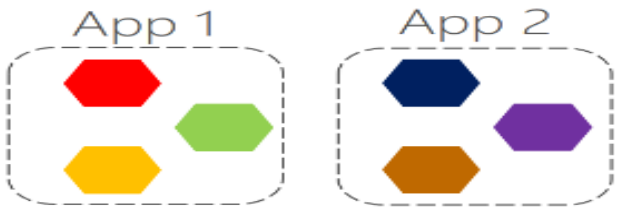
Microservices

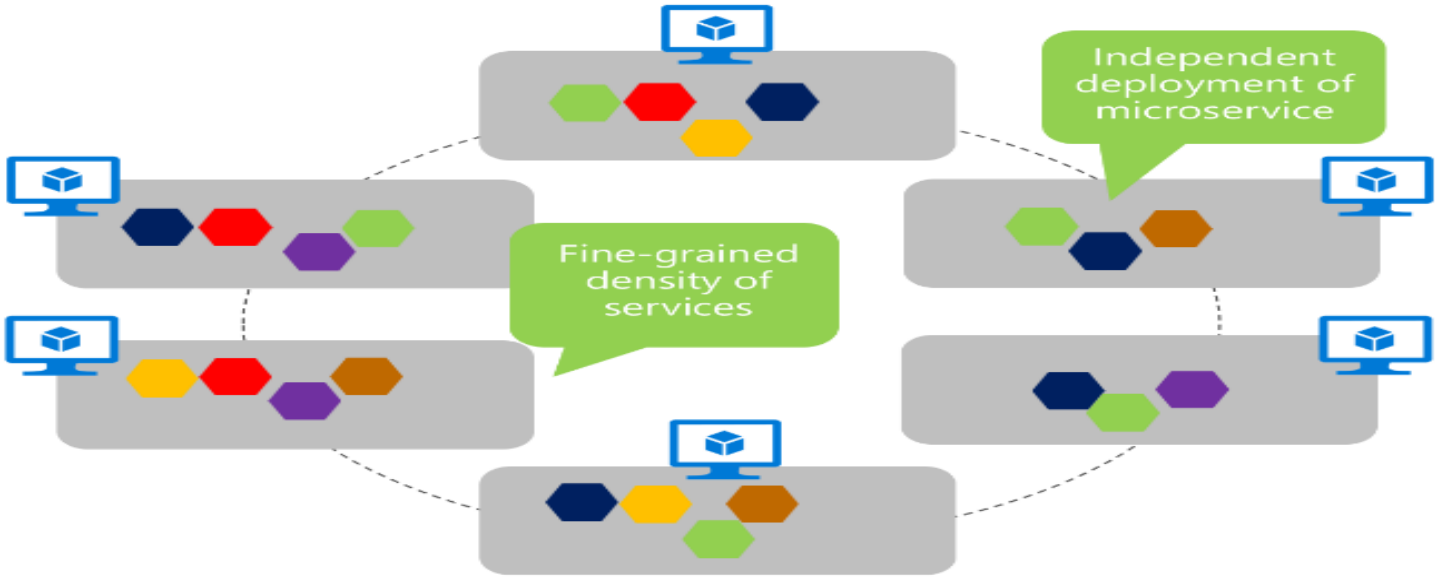
- Microservices can scale out independently. Instead of having a single monolithic application that you must scale out as a unit, you can instead scale out specific microservices.
- That way, you can scale just the functional area that needs more processing power or network bandwidth to support demand, rather than scaling out other areas of the application that don't need to be scaled. That means cost savings because you need less hardware.
- One important benefit of a microservices approach is that teams are driven more by customer scenarios than by technology. Smaller teams develop a microservice based on a customer scenario and use any technologies they want to use.

Microservices

Microservices application approach

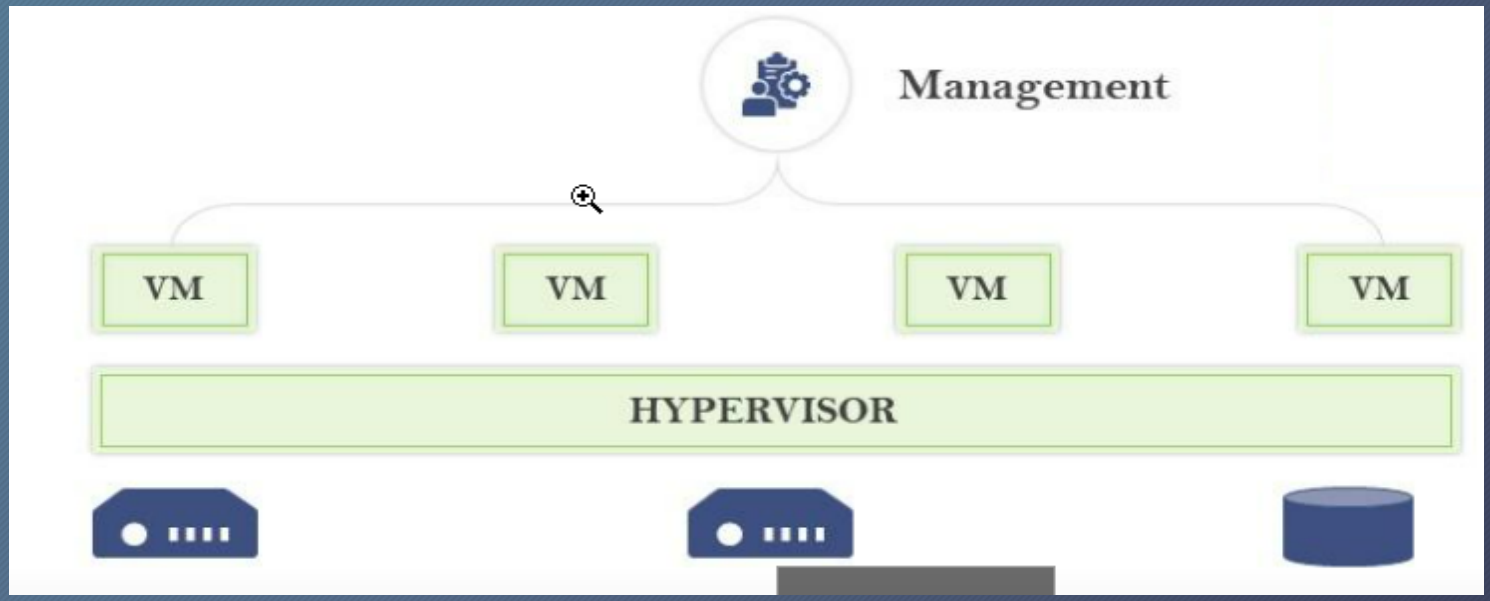
- A microservice application segregates functionality into separate smaller services.
- Scales out by **deploying each service independently** with multiple instances across servers/VMs





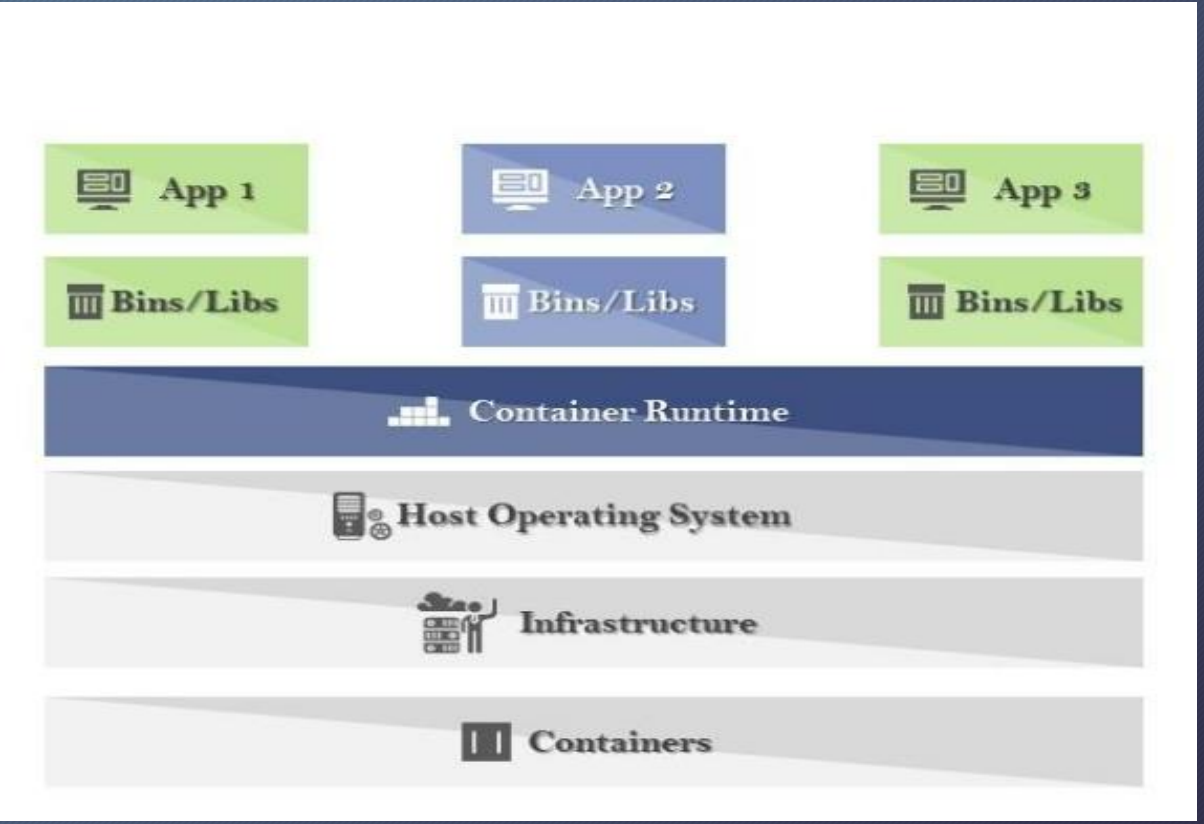
Virtualization

- Virtualization refers to the act of making a virtual version of one thing, together with virtual hardware platforms, storage devices, and electronic network resources.



Containerization

- Containerization is a form of virtualization where applications run in isolated user spaces, called containers, while using the same shared operating system (OS).



Multiplicity of Goods

Do I worry about how goods interact (e.g. coffee beans next to spices)

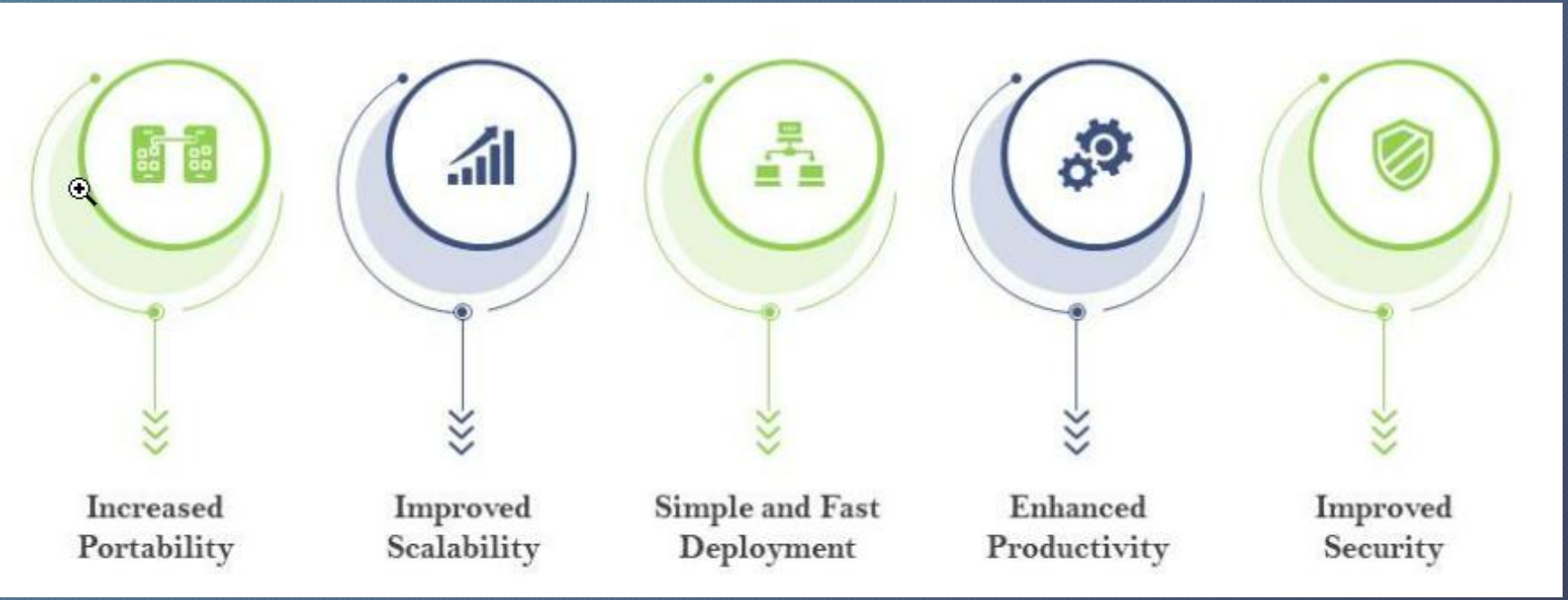
A standard container that is loaded with virtually any goods, and stays sealed until it reaches final delivery.

…in between, can be loaded and unloaded, stacked, transported efficiently over long distances, and transferred from one mode of transport to another

Multiplicity of methods for transporting/storing

Can I transport quickly and smoothly (e.g. from boat to train to truck)

Advantages of Containerization over Virtualization

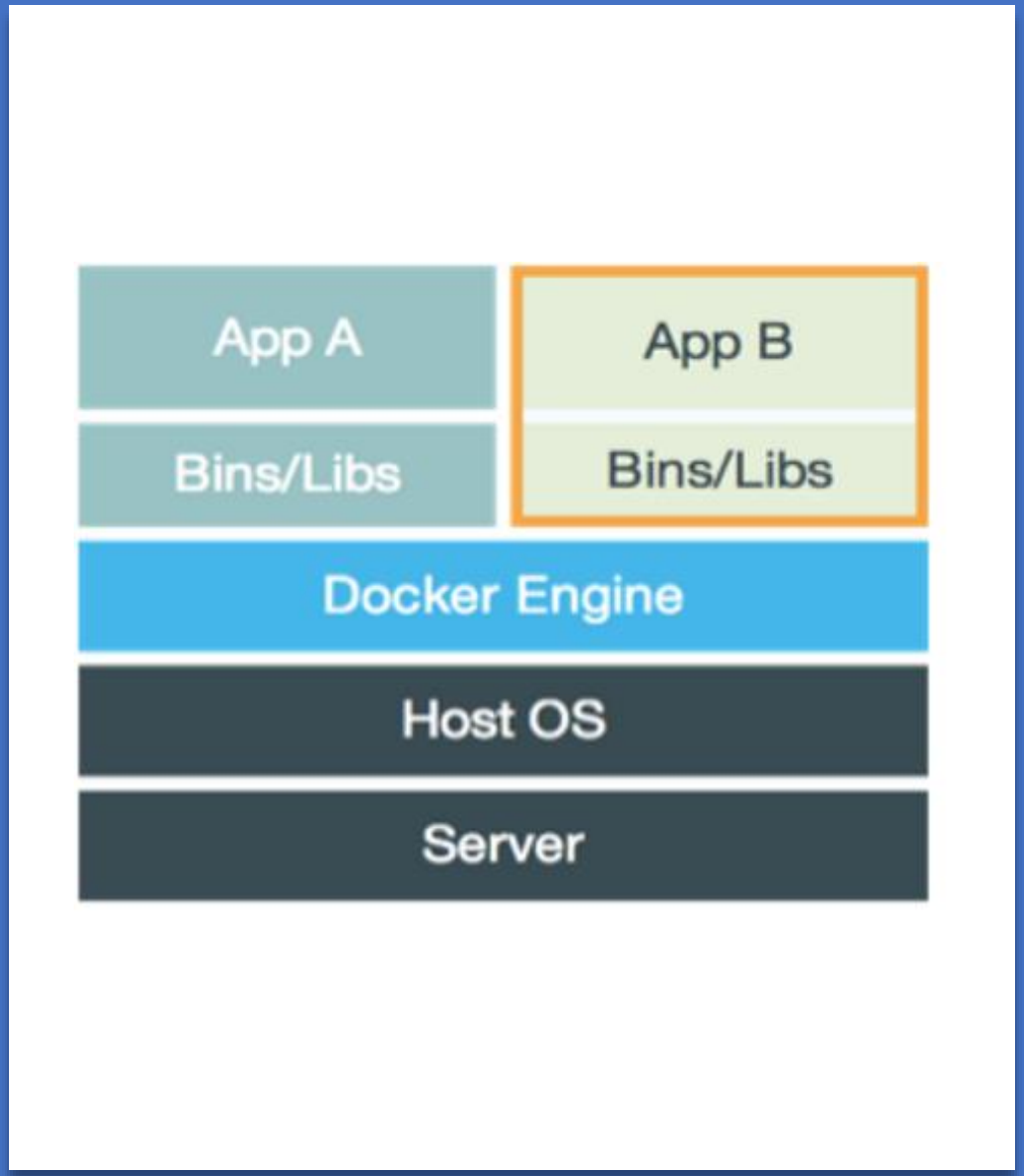


Containerization

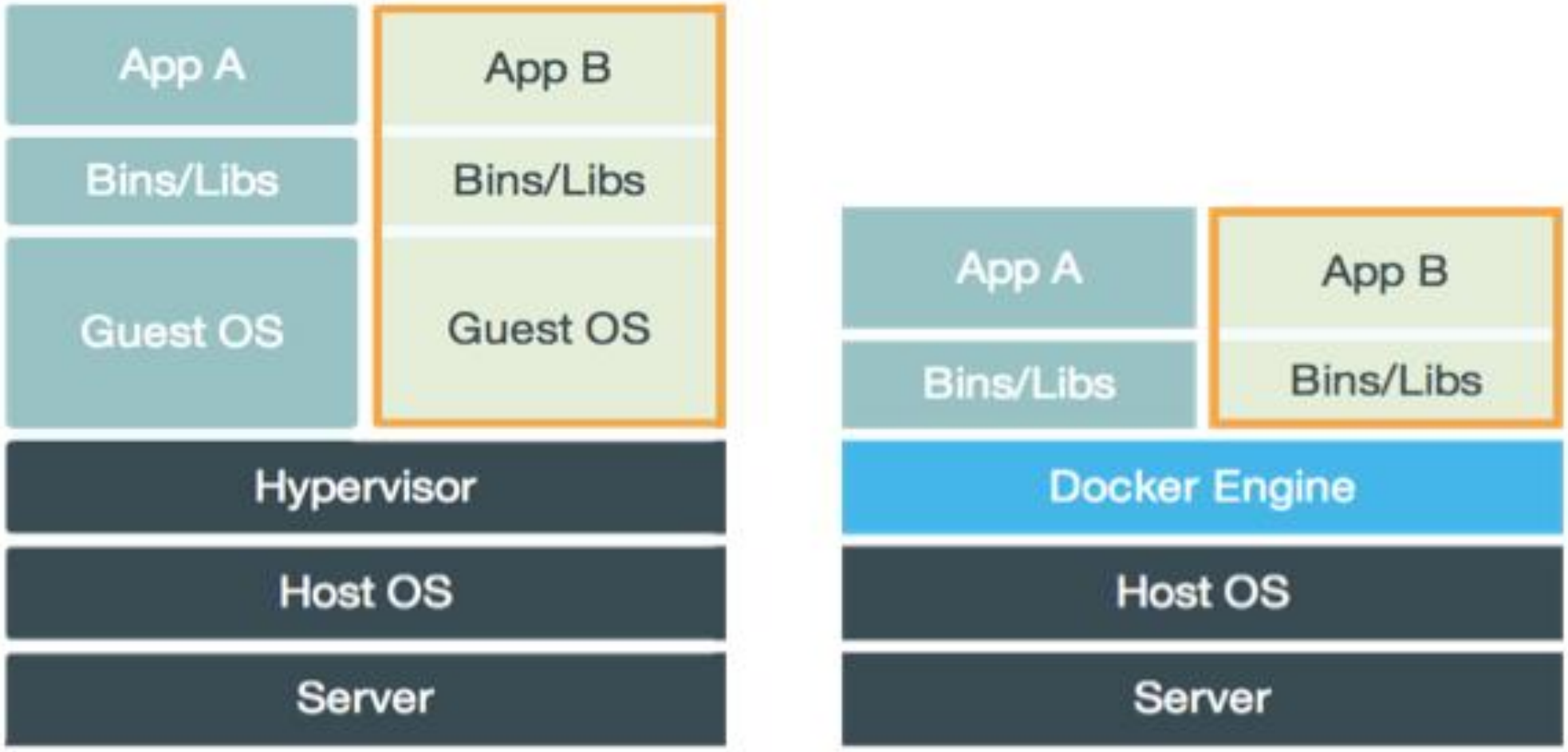
- Containerization is an approach to software development in which an application or service, its dependencies, and its configuration (abstracted as deployment manifest files) are packaged together as a container image. You can test the containerized application as a unit, and deploy them as a container image instance to the host operating system (OS).
- Just as shipping containers allow goods to be transported by ship, train, or truck regardless of the cargo inside, software containers act as a standard unit of software deployment that can contain different code and dependencies. Containerizing software this way lets developers and IT professionals deploy them across environments with little or no modification.

Docker

- Docker is an open-source project for automating the deployment of applications as portable, self-sufficient containers that can run in the cloud or on-premises.
- Docker containers can run anywhere on Azure: on-premises in the customer's datacenter, in an external service provider, or in the cloud. Docker image containers can run natively on Linux and Windows.
- Docker may be a set of platform as a service product that uses OS-level virtualization to deliver software package in packages knows as Container.



Virtualization Versus Docker Container



Image

- When a developer uses Docker, they create an app or service and package it and its dependencies into a container image.
- An image is a static representation of the app or service and its configuration and dependencies.
- It's this image that, when run, becomes our container. The container is the in-memory instance of an image.
- A container image is immutable. Once you've built an image, the image can't be changed.
- Since you can't change an image, if you need to make changes, you'll create a new image. This feature is our guarantee that the image we use in production is the same image used in development and QA.

Docker File

- A Dockerfile is a text file that contains instructions on how to build a Docker image.
- Dockerfiles are written in a minimal scripting language designed for building and configuring images. They also document the operations required to build an image starting with a base image.
- To create a Docker image containing your application, you'll typically begin by identifying a base image to which you add more files and configuration.
- The process of identifying a suitable base image usually starts with a search on Docker Hub for a ready-made image that already contains an application framework and all the utilities and tools of a Linux distribution like Ubuntu or Alpine.
- For example, if you have an ASP.NET Core application that you want to package into a container, Microsoft publishes an image called mcr.microsoft.com/dotnet/core/aspnet that already contains the ASP.NET Core runtime.

Demo – Docker File

- Step # 1: Clone Project
  - git clone https://github.com/MicrosoftDocs/mslearn-dotnetmicroservices

- Step # 2: Create Docker File in backend project
  - Open the backend directory from the repository that you cloned in a text editor such as VS Code.
  - Within the backend directory, open the file named Dockerfile; this file will be empty.
  - Enter the following code:
    - FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
    - WORKDIR /src
    - COPY backend.csproj .
    - RUN dotnet restore
    - COPY . .
    - RUN dotnet publish -c release -o /app

- This code will perform the following steps sequentially when invoked:
  - Pull the mcr.microsoft.com/dotnet/sdk:6.0 image and name the image build
  - Set the working directory within the image to /src
  - Copy the file named backend.csproj found locally to the /src directory that you created
  - Calls dotnet restore on the project
  - Copy everything in the local working directory to the image
  - Calls dotnet publish on the project

Demo – Docker File

- Step # 3: Clone Project
  - FROM mcr.microsoft.com/dotnet/aspnet:6.0
  - WORKDIR /app
  - EXPOSE 80
  - EXPOSE 443
  - COPY --from=build /app .
  - ENTRYPOINT ["dotnet", "backend.dll"]

- This code will perform the following steps sequentially when invoked:
  - Pull the mcr.microsoft.com/dotnet/aspnet:6.0 image
  - Set the working directory within the image to /app
  - Exposes port 80 and 443
  - Copy everything from the /app directory of the build image you created into the app directory of this image
  - Sets the entrypoint of this image to dotnet and passes backend.dll as an argument

Demo – Docker File

- Step # 4: Save Dockerfile
  - Save and open a command prompt and go to the directory that holds that file.
- Step # 5: Create Docker Image
  - Run this command to create the image and tag it with the name pizzabackend:
    - docker build -t pizzabackend .
- Step # 7: Run API
  - To run the web API service, run the following command to start a new Docker container using the pizzabackend image and expose the service on port 5200:
    - docker run -it --rm -p 5200:80 --name pizzabackendcontainer pizzabackend
  - Browse to http://localhost:5200/pizzainfo and see a JSON representation of Contoso Pizza's menu.

Microservices Orchestration

- As more services get added, the overall system becomes more complex to scale out and manage. Orchestrators can help.
- In a microservice-based approach, each microservice owns its model and data so it will be autonomous from a development and deployment point of view. These kinds of systems are complex to scale out and manage; therefore, you absolutely need an orchestrator if you want to have a production-ready and scalable multi-container application.
- The orchestrator helps with composing applications consisting of many microservices into one deployable unit. That unit is then moved — or deployed — to a host.

Microservices Orchestration

- The orchestrator helps with managing the host.
  - It can automatically start the containers, scale them out with multiple instances per image, suspend them, or shut them down when needed. The orchestrator can also control how containers access resources like the network and data storage.
- Orchestrators can perform tasks such as load-balancing and routing in scenarios when multiple containers exist on multiple hosts in a complex distributed system. They can also monitor the containers' and hosts' health.

Docker Compose

- It is a tool that build multiple Docker images together as a single unit and then deploy that unit.
- With Docker Compose, you can use a YAML file to configure an application's services.
- The Docker Compose tool then provides a way to build the individual services, and a means to start them.

Demo – Docker Compose

- To build the container images, open a command prompt and run the following command:
  - docker-compose build
- Then, to start both the website and the web API, run this command:
  - docker-compose up

Kubernetes

- Kubernetes is a portable, extensible open-source platform for managing and orchestrating containerized workloads.
- Kubernetes abstracts away complex container management tasks and provides you with declarative configuration to orchestrate containers in different computing environments.
- This orchestration platform gives you the same ease of use and flexibility you may already know from platform as a service (PaaS) or infrastructure as a service (IaaS) offerings.

Benefits of Kubernetes

- These tasks include:
  - Self-healing of containers. An example would be restarting containers that fail or replacing containers.
  - Scaling deployed container count up or down dynamically, based on demand.
  - Automating rolling updates and rollbacks of containers.
  - Managing storage.
  - Managing network traffic.
  - Storing and managing sensitive information, such as usernames and passwords.

Kubernetes

- In order for Kubernetes to create a container image, it needs a place to get it from. Docker Hub is a central place to upload Docker images. Many products, including Kubernetes, can create containers based on images in Docker Hub.

Demo – Kubernetes

- Step # 1 : Download repo
- Step # 2 : Build and run docker containers to verify
- Step # 3 : Sign in into Docker Hub
  - docker login
- Step # 4 : Upload Images to Docker hub
  - docker tag pizzafrontend [YOUR DOCKER USER NAME]/pizzafrontend
  - docker tag pizzabackend [YOUR DOCKER USER NAME]/pizzabackend

  - docker push [YOUR DOCKER USER NAME]/pizzafrontend
  - docker push [YOUR DOCKER USER NAME]/pizzabackend

Demo – Kubernetes

- Step # 5 : Create YML Files
- Step # 6 : Deploy and Run
  - kubectl apply -f backend-deploy.yml
  - This command is telling Kubernetes to run the file we created. It will download the image from Docker Hub and create the container.

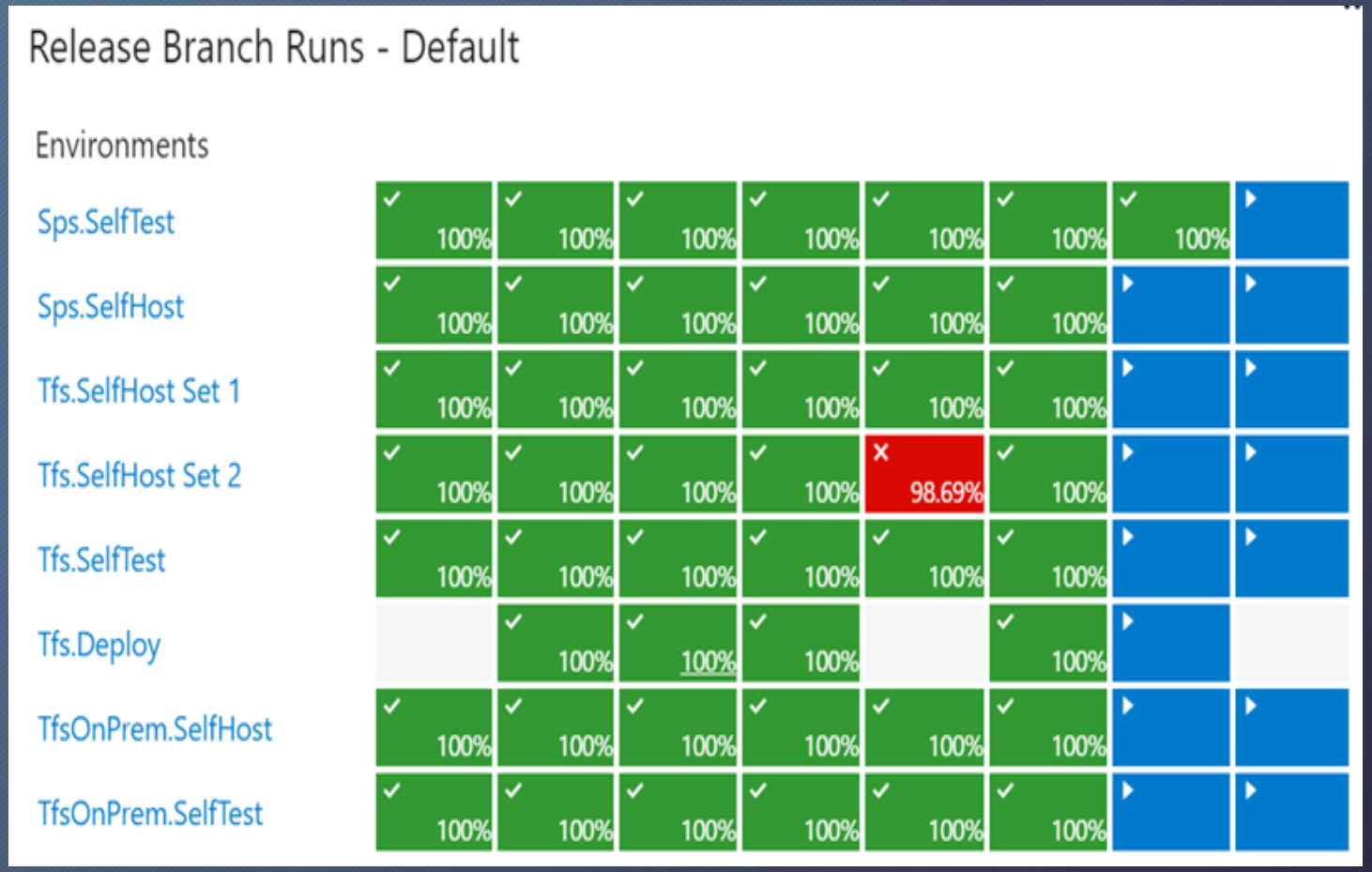
  - To view the progress, use the following code.
  - kubectl get pods
  - When everything is ready, there'll be a 1/1 under the READY column and Running under the STATUS column.
- Step # 7 : Scale Container
  - kubectl scale --replicas=5 deployment/pizzabackend

Release Process

- The quality of your release process can't be measured directly because it's a process.
- What you can measure is how well your process works.
  - If your release process constantly changes, it might indicate something wrong with the process.
  - If your releases continuously fail, and you regularly must update your release process to make it work, it might also suggest that something is wrong with your release process.
  - Maybe something is wrong with the schedule on which your release runs, and you notice that your release always fails on a particular day or at a specific time. Or your release always fails after the deployment to another environment. It might be an indication that some things are maybe dependent or related.
- You can keep track of your release process quality by creating visualizations about the quality of all the releases following that same release process or release pipeline.

Release Process

The release also has a quality aspect, but it's tightly related to the quality of the deployment and package deployed. When we want to measure the quality of a release itself, we can do all kinds of checks within the pipeline.



Release notes and Documentation

When you deploy a new release to a customer or install new software on your server, and you want to communicate what has been released to your customer, the usual way is to use release notes.

Where do the release notes come from?

- Document store
  - An often-used way of storing release notes is by creating text files or documents in some document store. This way, the release notes are stored together with other documents.
  - The downside of this approach is that there's no direct connection between the release in the release management tool and the release notes that belong to this release.

- Wiki
  - The most used way for customers is to store the release notes in a Wiki. For example:
    - Confluence from Atlassian
    - SharePoint Wiki
    - SlimWiki
    - Wiki in Azure DevOps
  - Release notes are created as a page in the wiki and by using hyperlinks. Relations can be associated with the build, the release, and the artifacts.

Release notes and Documentation

- In the codebase
  - When you look at it, release notes belong strictly to the release of the features you implemented and your code. In that case, the best option might be to store release notes as part of your code repository.
  - Once the team completes a feature, they or the product owner also write the release notes and save them alongside the code. This way, it becomes living documentation because the notes change with the rest of the code.

- In a work item
  - Work items can be Bugs, Tasks, Product Backlog Items, or User Stories.
  - You can create or use a different field within the work item to save release notes in work items. In this field, you type the publicly available release notes that will be communicated to the customer.
  - With a script or specific task in your build and release pipeline, you can generate the release notes and store them as artifacts or publish them to an internal or external website.

Release notes and Documentation

- Creating a wiki is a better and more modern way to store your documentation.
  - Wiki's contain only text files called **Markdown Files**.
  - These markdowns can refer to pictures, hold code samples, and be part of your code repository.
  - Changes and history can be easily tracked.
  - Wiki is accessible to everyone in your team. People can work together on the documentation instead of waiting for each other.
  - Manuals or documentation you release together with the product should be treated as source code.
  - When the product changes and new functionality are added, the documentation needs to change.
  - Create a documentation artifact in your build pipeline and deliver this artifact to the release pipeline.
  - The release pipeline can then deploy the documentation to a site or include it in the boxed product.