CONTAINERS

•••

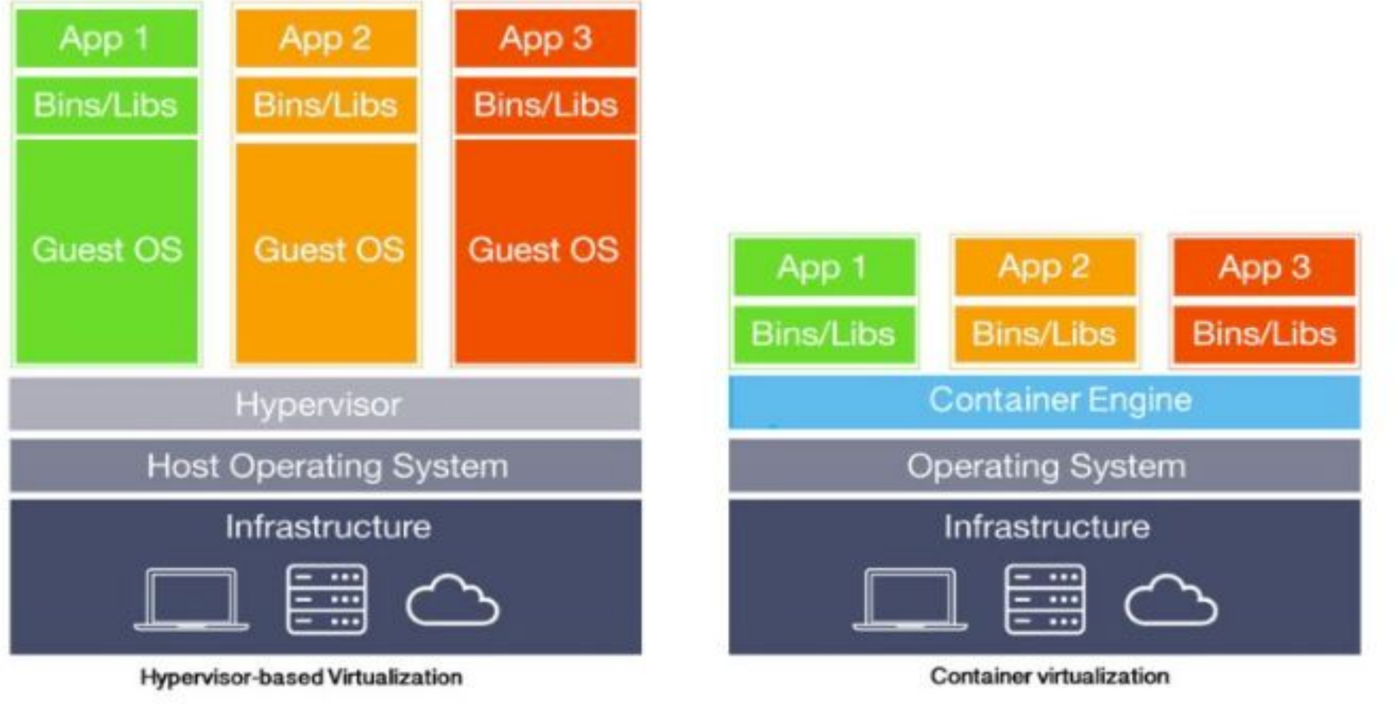
INTRODUCTION TO CONTAINERS

- **Container technology**, also known as just a **container**, is a method to package an application.
- Any application can be bundled in a container can run without any worries about dependencies, libraries and binaries.
- Container creates the isolated environment with all the required dependencies, libraries and binaries to run your application without any issue.
- The application can run in any environment.

INTRODUCTION TO CONTAINERS

- A container is a standard unit of software that packages up a given code and all its dependencies so the application runs quickly and reliably from one computing environment to another.
- **Containerization** is a lightweight alternative to a virtual machine that involves encapsulating an application in a container with its own operating system.
- Containerization is the process of bundling your application code with requires packages/libraries required at runtime to execute your application quickly and reliably in any supported computing environment

Containers VS VM:



CONTAINERS VS VM

- Each virtual machine runs a unique guest operating system
- Each VM has its own binaries, libraries, and applications
- **Container** systems usually provide service isolation between containers.
- Containers provide a way to run these isolated systems on a single server or host OS.
- Containers sit on top of a physical server and its host OS. Containers are only megabytes in size and take just seconds to start, not like VM.

INTRODUCTION TO CONTAINERS

Monolithic applications are proved to be hard maintained, maintaining and CI/CD of such applications is time and energy intensive.

Containerization offers the following benefits:

- Portability of distributed applications
- Reproducibility of the application
- Scaling based on requirements
- Lifecycle management of containers
- Memory, CPU, and storage efficiency compared to VM hosting and hence cluster improvisation

Container Images:

- Docker an open source project, generated the most interest in container technology in the past few years.
- A command line tool that made creating and working with containers easy for developers and administrators.
- A container **image** is an inert, immutable, file that’s essentially a binary packaged snapshot of a **container**.
- An image is the application we want to run.
- A Container is an instance of that image running as a process.

Docker Installation

- **Official Ubuntu Repositories**

  -
    ```
    $ sudo apt-get install docker.io
    ```
- **Another Way TO install Docker from Official Site**
  - https://docs.docker.com/install/linux/docker-ce/ubuntu/
- **Verify the installation**

  -
    ```
    $ sudo docker -v
    ```

DOCKER:

- **Verify that Docker CE is installed correctly by running the hello-world image**.

  -
    ```
    $ sudo docker run hello-world
    ```
- **CHECK IMAGES:**

  -
    ```
    $ sudo docker image ls
    ```

To Build Docker Image:

Create index.html file

```
<!doctype html>
<html>
 <head>
 </head>
 <body>
   <p>Hello EveryOne.</p>
 </body>
</html>
```

To Build Docker Image:

- **Create Dockerfile**
  - Add the following Instructions in Dockerfile:
  - **FROM** instruction to set the application's base image.
    - FROM nginx:alpine
  - **COPY** files from a specific location into a **Docker** image.
    - COPY index.html /usr/share/nginx/html/index.html

To Build Docker Image:

- **BUILD IMAGE:**
  - The "-t" flag adds a tag to the image so that it gets a nice name and tag.
  - At the end in the below command “.” which tells Docker to use the

    Dockerfile in the current directory.
    - docker build -t <image-name> .

DOCKER

- **CHECK IMAGES AGAIN:**

  -
    ```
    $ sudo docker image ls
    ```
- **Run IMAGE:**
  - Get Image Name from above command.

    -
      ```
      $ sudo docker run -p=8080:80 <image-name>
      ```

DOCKER HUB AND ADVANCED COMMAND WILL UPDATE BY TOMORROW THANKS