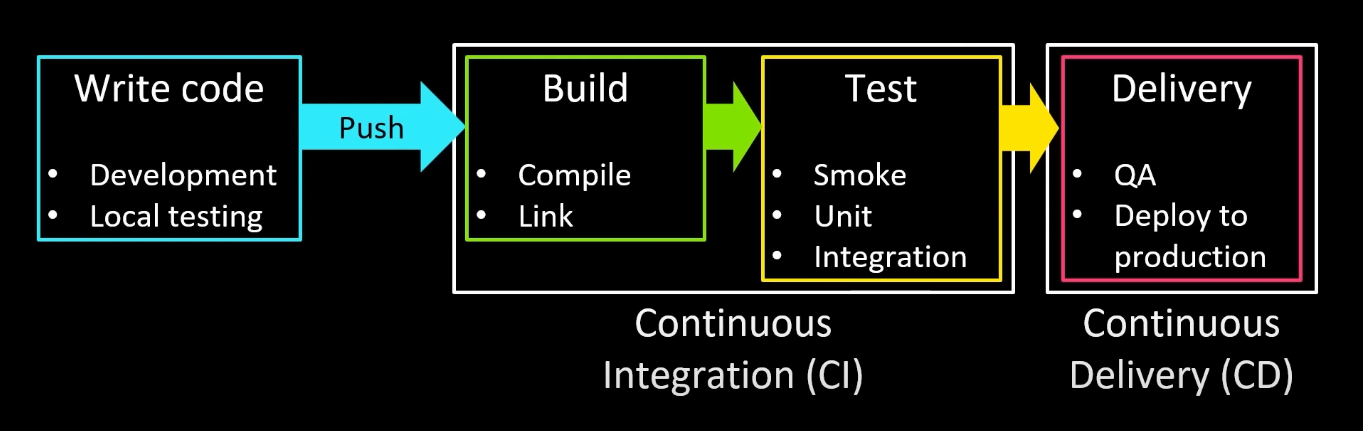
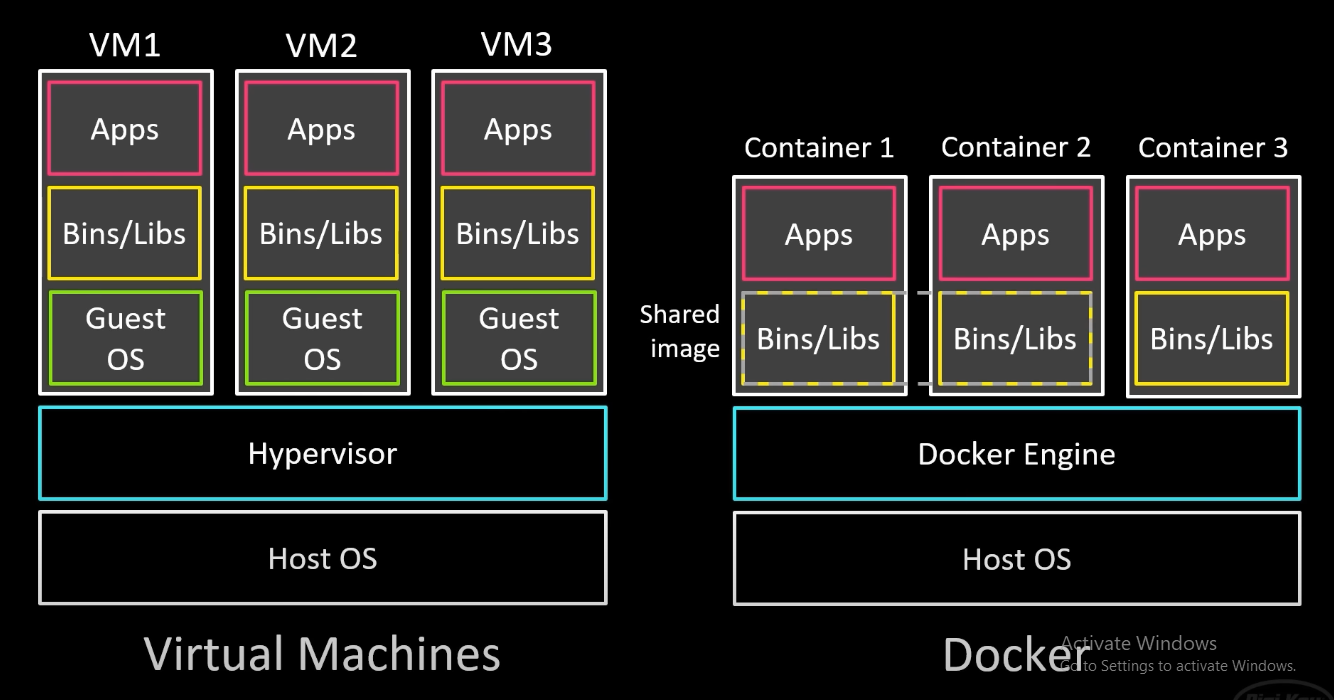
**Docker for Embedded**

1. **CI/CD –** continuous Integration/ continuous deployment

* Process of automating the testing and deployment of a software project.
* CICD pipeline****

1. **Docker Intro:**

* Docker is a tool that lets you package your code and all its dependencies into a single container so it runs the same everywhere, on any computer.
* Docker is a platform that runs applications in virtualized instances known as “containers.” Containers use a shared set of binaries and libraries as well as low-level drivers from the host operating system (OS) to run applications. As a result, applications can be made portable and scalable, regardless of the host OS.

1. **Docker Installation and setup:**

* Enter <https://www.docker.com/> and click **Download Docker Desktop** ( If you have an AMD or Intel processor (x64), you should download the AMD64 version. If you have an ARM-based processor, you should download the ARM64 version)

1. **Creating a simple Docker project**

* Create a project folder wherever necessary “hello\_docker” and $cd hello\_docker
* Create a **Dockerfile** and open it in any Editor(vs code)

# Fetch ubuntu image

FROM ubuntu:22.04

# Install Python on image

RUN \

    apt-get update && \

    apt-get install -y python3 && \

    apt-get install -y build-essential

# Create a directory for our tests

RUN mkdir /tests

# Copy in our Python script

COPY test.py /tests/test.py

# Copy in our program under test

COPY main.c /tests/main.c

# Command that will be invoked when container starts

ENTRYPOINT ["python3", "tests/test.py"]

* After creating Dockerfile in the same directory, open cmd or terminal and run the command **> docker build -t my-image .**
* To listout all built images **> docker images** and you’ll be able find the created image in docker desktop -> images
* Create new Container using command **> docker create -i -t --entrypoint="/bin/bash" --name my-container my-image**
* *-i* means “interactive mode,” which keeps the container running to allow for interactions.
* *-t* means “tty” to add a pseudo terminal for command line interactions with the container.
* *--entrypoint* overrides the entrypoint in the Dockerfile. We do this for this example so you can see how to log in to a container without running the test.py script.
* *--name* gives a name to the container. If you don’t specify a name, Docker will randomly assign a name by default.
* To list all containers **> docker container ls -a**
* To start a container **>** **docker start -i my-container**
* To add another file test2.py **> docker cp test2.py my-container:/tests/test2.py**
* Now to start our container > docker **start** -i my-**container**
* To get out of the container(my-container) type **>exit**
* To remove container **> docker rm my-container**
* To remove image **> docker rmi my-image**
* To run a container from the image my\_image, and automatically clean it up after it stops **>docker run –rm my\_image**
* Reference
  + Docker images: https://hub.docker.com/
  + Dockerfile: <https://docs.docker.com/reference/dockerfile/>
  + Youtube: <https://youtube.com/playlist?list=PLEBQazB0HUyR00bgoctk-5uZoOrbmPASC&si=SDyVC7qhEzAGdHvi>