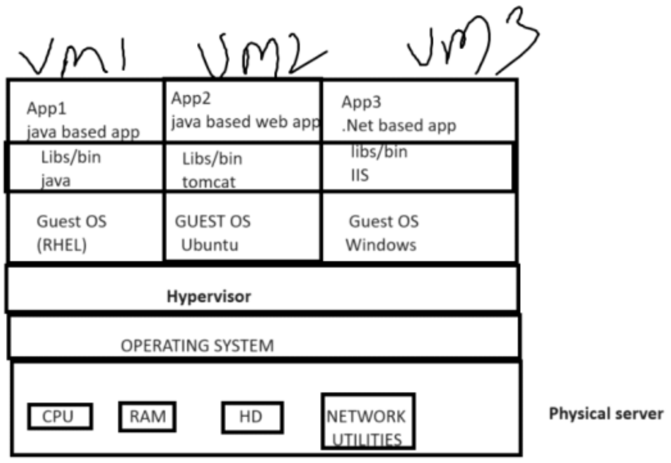
Docker-Day-1

14 December 2023 10:12

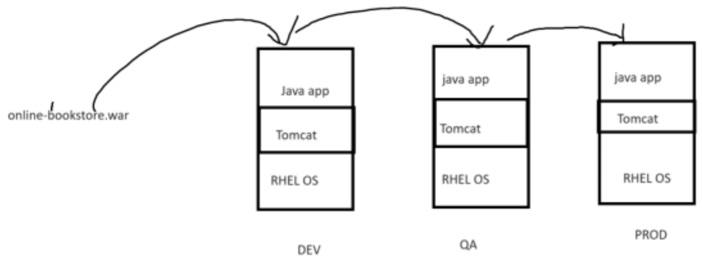
- **What are VM's?**
  The EC2 instances that we created on AWS management console are considered as VM's

- **How these VM's are created?**
  - We just go to the AWS management console & click launch instance, correct right? - Yes
  - But in background we will discuss what will happen,



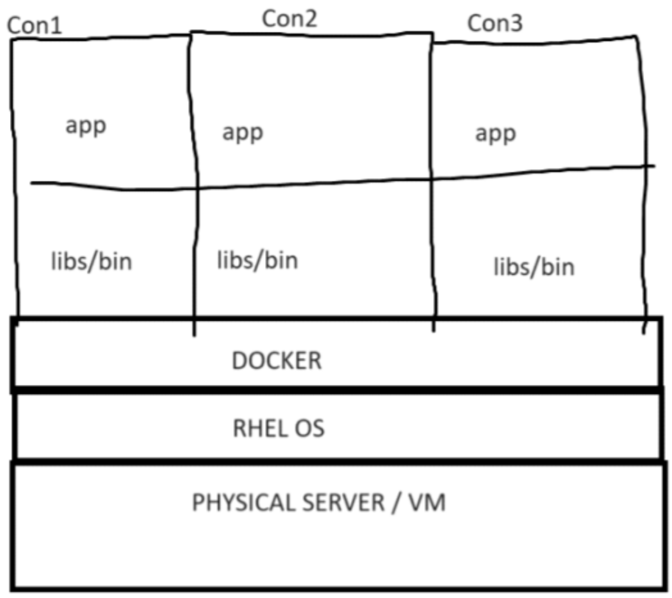
- Assume there is **physical server** with Linux OS on datacentre with good amount of
  - CPU
  - RAM
  - HD
  - Network utilities
- Whatever machine that physically present it can be considered as physical server.
  - I mean laptop/any computers
- On top physical server Hypervisor software is installed.
- **What Hypervisor software will do?** This helps to create virtual machines.
- On these virtual we can install OS like
  - RHEL
  - Ubuntu
  - Windows
- On top of these OS we will install s/w's (or) binaries (or) libraries, assume on
  - VM1 we installed Tomcat
  - VM2 we installed Java
  - VM3 we installed IIS
- Post that we deploy applications on these VM's
  - VM1 ==> Tomcat ==> War
  - VM2 ==> Java ==> .jar
  - VM3 ==> IIS ==> .Net application

- **What is the problem that may happened with VM's?**
  - **In real time, how will bring any application to the live environment?**
    First, we will deploy application into DEV environment & next we will deploy the application into QA environment & finally we will deploy application into production environment, Correct or not? - Correct



- **Assume you have a Java web application called Online Bookstore and you want to take it live.**
  - Once the developers completed the coding part, as a DevOps engineer will generate a binary file called **online-bookstore.war** file.
    This file we will deploy into DEV environment VM first.
  - **To deploy java application on DEV environment what is required?**
    We will create a VM with Linux OS on top of that tomcat software & finally we will deploy the Java application.
    So, post deployment the developers will do the testing & confirmed the application is working fine in DEV environment & asked to promote into QA environment.
  - As a DevOps engineer again, we will create a VM to the QA environment with Linux OS, tomcat software & finally deploy applica tion into QA environment.
  - Here again, the QA engineer will do the testing.
    During QA testing identified a lot many issues, so simply QA engineer will create a bug in the JIRA & will assign to develope rs.
  - Now the developers will replies, there is no issues in the code at their side & the application also working fine in the dev environment. This might be configuration issues at QA VM itself.
  - Now who did this configuration of VM like software installation, and deployment? DEVOPS ENGINEER
  - So devops engineer will check all software's are installed or not in QA VM & since all software's installed simply revered th e mail/ticket to DEVELOPERS there are no configuration issues at QA VM side.
  - Like this conversation happened between developers & ops person for couple of days.
  - One fine day the was issue identified, the tomcat version is different in both servers.
    Due to this environment mismatching some of the future not working in the QA environment which are working in the development .
  - Finally time got wasted this kind of environmental issues.
  - To avoid this kind of situations containers came into picture.

► **What containers will do?**
- Containers will have the software's that are needed to run your application & Artifact of the application code.
- Now the probability for getting errors related to the environment changes between different environments very less, since we packed the .war file & tomcat software. Simply we can container on any environment & can access application.
- Now let's see how to create containers in background.



- Assume this is a physical server or VM with good amount of RAM, CPU and hard disk on top of that
  - RHEL was installed & again on top of that docker is installed. So, this **docker software what it will do, it will help you create the containers.**
- Let's create containers
  - container-1
  - container-2
  - container-3
- Assume on these containers we install libs/bin's & finally we deploy our application.
  - On container-1 we can install tomcat binary & deploy java web application .war file.
  - On container-2 we can install java binary & deploy java application .jar file

        - On container-3 we can install IIS & deploy .Net application .jar file

    - These containers are independent each other & we can create n-no of containers on single VM.

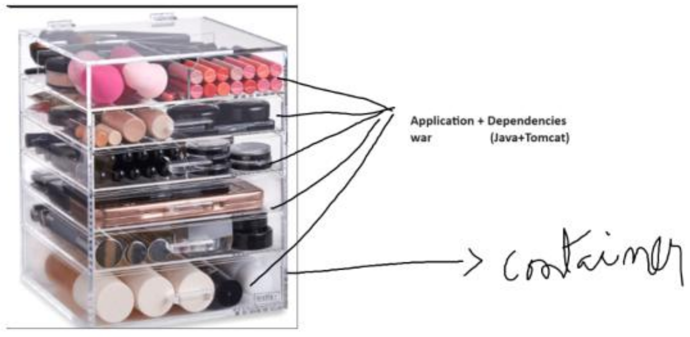
    - **You may have doubt how containers will work with-out OS?**
      Container will communicate directly with the OS of physically as read only.

    - For your easy understanding following are containers in real-time,
        - Shipping container

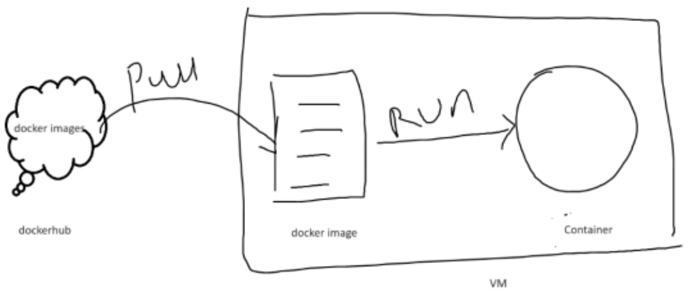
        - Lunch box

        - Makeup box

    - Here containers like the boxes for us & the **items that present in boxes**/containers are considered as **applications & dependencies needed** for project.



- **Now let's see how to containers practically.**



    - To create any docker container we require **docker image**.

- **What is docker image?**
  - It's a read only template which contains the list of instructions to create the container.
    - Suppose you want to create Ubuntu OS container you can find Ubuntu image &
    - Same way if you want to create tomcat container we need tomcat docker image.

  - **Where do this docker images stored?**
    - That docker images will stored under the registry called Docker Hub.
    - In the Docker hub we can find docker images for any software.
      - Let's search for **Ubuntu** docker image here we can find n-no.of docker images created by different people.
      - Same way if you search **jenkins** you can find jenkins docker image details
      - Out of these docker images we will only the **Official** Image which is maintained docker community.
      - You may have doubt what are the rest of the docker images?
        - Those are docker image created by the **people like us & pushed into Docker hub**. We see this part on next sessions.

- How to list the active containers that running in VM
  docker ps
  Since we don't have any active docker container we don't see any output.

- Now the run the **docker ps -a**
  This command will help to list the inactive & active containers, present we don't have any active/inactive container. So the output is nothing.

- Let's **Create Ubuntu container on docker**, for that first we need to
  - **Download Ubuntu docker image from docker hub** to the server, for that we will go Docker hub & look for docker image Ubuntu
    Next we will run
    docker pull ubuntu

    If I don't mention anything after ubuntu by default latest docker image will be picked & if we mention specific tag particular docker image will be downloaded.

  - Based on this output we can confirm docker image is downloaded but to confirm what docker images available or not on the server run command
    docker images

  - Ubuntu image is ready on server & we can create Ubuntu container based on this by running command
    docker run ubuntu

  - Now we can check container is running or not with the command
    docker ps

- **To connect actively running container**
  docker exec -it <cid> /bin/bash

  - Only we can login to the OS related containers.

  - Now we are inside the container & here we can see the
    - login user on VM was ec2-user but now we are under root
    - file system structure is also different now.

  - To come-out this container we can enter a command exit, now we are on the server.

  - **If you run docker ps command we don't see any docker containers, what happened?**
    After click on exit in docker container, it will be automatically stopped. You can find stopped container using
    docker ps -a

- **If you want to connect container & come outside of it without stopping container we will see now.**

  - Create new docker container from Ubuntu image
    docker run Ubuntu

  - Connect to docker container
    docker exec -it <cid> /bin/bash

  - To exit from container type
    Ctrl + p +q

- We can stop actively running container in docker using
  docker stop <cid>

- We can also start container by running
  docker start <cid>

- To remove the container from VM
  docker rm <cid>

- If you want to delete actively run command
  docker rm -f <cid>

- Same way if you want to remove the docker images present on server
  docker rmi <image_name>

- If you want to remove all the docker images
  docker rmi -f (docker images -q)