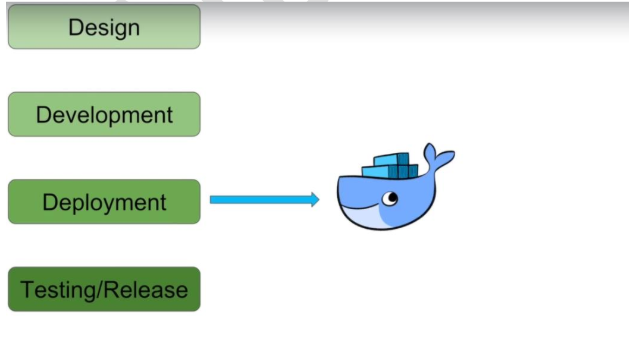
 **DOCKER TUTORIAL – 4**

**What covered in previous lecture:**

* **What is docker ?**
* **Why docker**
* **What is containerisation**
* **Docker setup in linux ubuntu ec2**
* **What is Dockerfile**
* **Dockerfile Keywords**
* **DockerHub account**
* **Dockerhub Integration with VMs -Docker login**
* **Dockerhub push & pull**
* **Working with Docker Images**

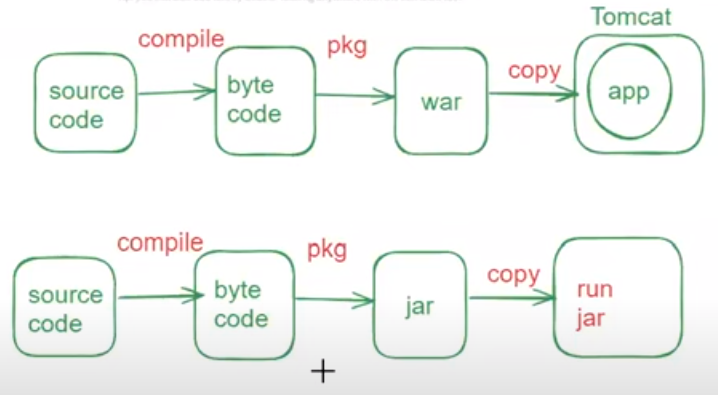
**What you will Learn in this Tutorial:**

* **Audience No knowledge of Container**
* **Developer Should know Docker**
* **Costume Dockerfile name – Dockerfile.aseem - Docker.dev**
* **Java Dockerfile**
* **Next.js Dockerfile**
* **Python app with Docker**
* **Docker Commands**
* **Create & Containerize your first (Lab)**
* **Docker File – Docker Image**
* **Overview of Docker Hub**

****

**Docker overview**

**Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By 5taking advantage of Docker's methodologies for shipping, testing, and deploying code, you can significantly reduce the delay between writing code and running it in production.**



[**https://github.com/Aseemakram19/maven-app.git**](https://github.com/Aseemakram19/maven-app.git)

[GitHub - Aseemakram19/maven-app](https://github.com/Aseemakram19/maven-app)

**Dockerfile for JAVA application (no springboot)**

**# Use a specific version of Tomcat as base image**

**FROM tomcat:8.0.20-jre8**

**MAINTAINER <aseem@gmail.com>**

**# Expose port 8080 to access the application**

**EXPOSE 8080**

**# Copy the WAR file from the target directory of your Maven project to the Tomcat webapps directory**

**COPY target/maven-cloudaseem-app.war /usr/local/tomcat/webapps/**

**Dockerfile for JAVA application (springboot)**

**# Use a specific version of Tomcat as base image**

**FROM openjdk:8**

**MAINTAINER <aseem@gmail.com>**

**COPY target/sbapp.war /usr/app/**

**WORKDIR /usr/app**

**EXPOSE 8080**

**ENTRYPOINT [“java”, “-jar”, “sbapp.jar”]**

**# Use an official Node.js runtime as a parent image**

**FROM node:14**

**# Set the working directory in the container**

**WORKDIR /usr/src/app**

**# Copy package.json and package-lock.json to the container**

**COPY package\*.json ./**

**# Install application dependencies**

**RUN npm install**

**# Copy the rest of the application files to the container**

**COPY . .**

**# Build the React app**

**RUN npm run build**

**# Expose port 3000 (assuming your Next.js app runs on port 3000)**

**EXPOSE 3000**

**# Define the command to run your application in development mode**

**CMD ["npm", "run", "dev"]**

**The Docker daemon**

**The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.**

**The Docker client**

**The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.**

**Docker registries**

**A Docker registry stores Docker images. Docker Hub is a public registry that anyone can use, and Docker looks for images on Docker Hub by default. You can even run your own private registry.**

**When you use the docker pull or docker run commands, Docker pulls the required images from your configured registry. When you use the docker push command, Docker pushes your image to your configured registry.**

**A container is defined by its image as well as any configuration options you provide to it when you create or start it. When a container is removed, any changes to its state that aren't stored in persistent storage disappear.**

**Example docker run command**

**The following command runs an ubuntu container, attaches interactively to your local command-line session, and runs /bin/bash.**

**docker run -i -t ubuntu /bin/bash**

**Access vm and**

**Docker components**

* **Docker image: Contains OS(very small)(almost negligible) + softwares**
* **Docker Container: Container like a machine which is created from Docker image.**
* **Docker file: Describes steps to create a docker image.**
* **Docker hub/registry: Stores all docker images publicly.**
* **Docker daemon: Docker service**

**Ways to create Docker Images**

* **Take image from Docker hub**
* **Create image from existing docker containers**
* **Create image from docker file**

**Dockerfile**

**Dockerfile :**

** A text file with instructions to build image**

**Docker HUB**

**Docker Hub is a cloud-based repository service where users can create, test, store, and distribute Docker container images.**

**It serves as a centralized platform for container image discovery and sharing, allowing developers to automate their build, test, and deployment pipelines with a vast library of trusted and user-contributed images. Docker Hub supports both public and private repositories, integrates with CI/CD tools, and provides features such as automated builds, webhooks, and image tagging.**

**LAB**

**1. AWS Account - Free**

**2. EC2 Create Ubuntu**

**3. Docker installation**

**4. Code App – github repo**

**5. Docker File**

**6. Docker Images**

**7. Docker Container**

**Permissions**

**sudo apt install docker.io -y**

**sudo usermod -aG docker ubuntu**

**sudo usermod -aG docker jenkins**

**groups jenkins**

**sudo chown ubuntu:docker /var/run/docker.sock**

**sudo chmod 660 /var/run/docker.sock**

**sudo systemctl restart jenkins**

**groups jenkins**

**sudo chown root:docker /var/run/docker.sock**

**sudo chmod 660 /var/run/docker.sock**

**sudo systemctl start docker**

**sudo systemctl enable docker**

**sudo docker build . -t flask-app:latest**

**sudo docker run -d -p 7000:7000 flask-app:latest**

**Docker Basic Commands**

**1. List Running Containers**

**docker ps**

**This command lists all currently running containers.**

1. **Check Docker Version**

**docker --version**

**This command checks the version of Docker installed on your system.**

1. **Get Detailed System-Wide Information**

**docker info**

**This command provides detailed information about the Docker installation, including the number of containers, images, storage driver, kernel version, operating system, and more.**

1. **Docker service status**

* **systemctl status docker**
* **systemclt start docker**
* **systemctl stop docker**

1. **Search an Image in local and Docker Hub Repository**

**docker search ubuntu**

1. **Pull an Image from Docker Hub**

**docker pull <image\_name>**

**For example, to pull the latest Ubuntu image:**

**docker pull ubuntu:latest**

1. **List Docker Images**

**docker images**

**This command lists all Docker images available on your local machine.**

1. **Run a Docker Container**

**docker run -it <image\_name> /bin/bash**

**For example, to run a container with the Ubuntu image:**

**docker run -it ubuntu /bin/bash**

**The -it flags keep the container interactive.**

1. **List All Containers**

**docker ps -a**

**This command lists all containers, including those that are stopped.**

1. **Stop a Running Container**

**You can get the <container\_id> from the docker ps command.**

**docker stop <container\_id>**

1. **Remove a Container**

**docker rm <container\_id>**

1. **To kill all running containers:**

**Docker kill $ (docker ps -q)**

1. **To delete all stopped containers:**

**docker rm $(docker ps -a -q)**

1. **Use the -f flag to force remove a running container:**

**docker rm -f <container\_id>**

1. **Remove an Image**

**docker rmi <image\_id>**

**You can get the <image\_id> from the docker images command.**

1. **To delete all images:**

**docker rmi $(docker images -q)**

1. **Build an Image from a Dockerfile**

**docker build -t <image\_name> .**

**The -t flag tags the image with a name. The . specifies the current directory containing the Dockerfile.**

1. **View Container Logs**

**docker logs <container\_id>**

1. **Execute a Command in a Running Container**

**docker exec -it <container\_id> <command>**

1. **For example, to open a bash shell in a running container:**

**docker exec -it <container\_id> /bin/bash**

