AGILE Methodology

During Development : sprints

Post Development : CI/CD

Continous Integration

Continous Deployment

Maven

Jenkins

Docker

Maven

1. Manage Dependency
2. Uniform/std project structure
3. Build (Package)
4. Test
5. Documentation
6. Reporting
7. Distribution

plugin in IDE

install maven

Maven batch : mvnw

CLI

Path variable for Maven

M2_HOME : Home to Maven installation folder

M2: Home to Maven CLI

Path Variable

POM.XML

- Inbuilt/details/parent POM.XML
  - default config file
- Custom POM.XML
- POM.XML (Effective) : Parent + Custom

Maven is plugin based tool

Maven CLI

```
>mvn <task/goal> [option]
```

For every goal we need a plugin

std/official maven plugin + third party plugin

```
> mvn <goal> : goes and look for  appropriate plugin from pom.xml file/installation folder
> mvn <plugin>:<goal>
```

```
>mvn archetype:generate -DgroupId=com.wf.training  -DartifactId=maven-demo
```

Scope of dependency

When that dependency/API would be needed in the lifecycle of project

build/compile

test

runtime

compile scope : (default)

build, test, run

provided scope

build, test, run (should not be package/exported)

Runtime env will provided

runtime scope

test and run

test

test

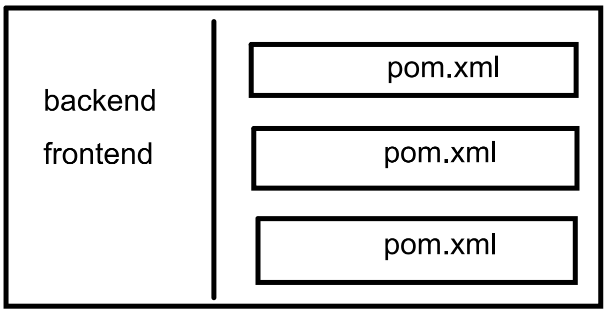
system scope

~ provided

build, test, run (not to be exported : runtime env will also not provide it)

explicit location is required to be mentioned , so that it will be downloaded at runtime on the fly

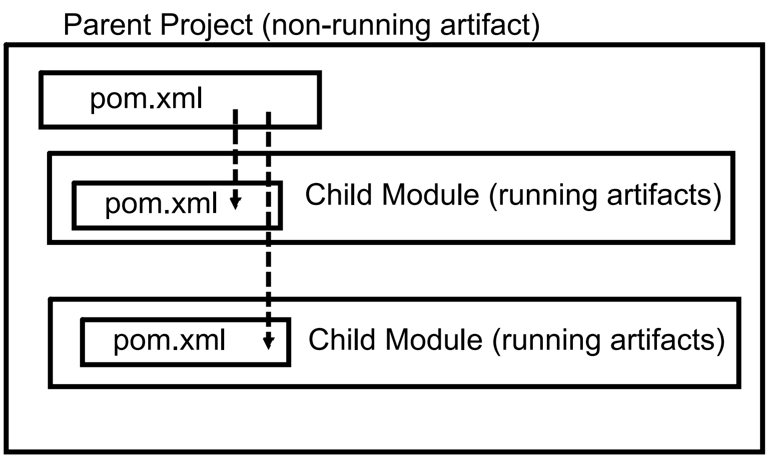
Multi Module Project



inheritance + Aggregation

Technically independent projects representing logically a single project are often defined as multi-module project

Parent Project (non-running artifact)
- child Project (running artifacts)
- child Project (running artifacts)


Parent Project (non-running artifact)
pom.xml
pom.xml
Child Module (running artifacts)
pom.xml
Child Module (running artifacts)

Inheritance

All the common dependency,

plugins

config can be placed in parent project pom.xml which can be inherited to child modules

Aggregation : any maven goal performed on parent will trigger same goals in all sub modules

Creating a simple java project

```
mvn archetype:generate -DgroupId=com.wf.training  -DartifactId=parent-app -
DarchetypeArtifactId=maven-archetype-quickstart
```

replace <packaging>jar</packaging> with <packaging>pom</packaging>

declare it as parent project/aggregator

Aggregation

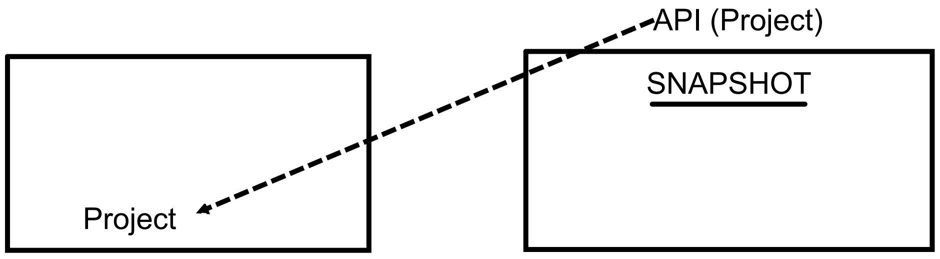
```
<modules>
   <module>child1-app</module>
   <module>child2-app</module>
   <module>child3-app</module>
 </modules>
```

Inheritance

```
<parent>
   <groupId>com.wf.training</groupId>
   <artifactId>parent-app</artifactId>
   <version>1.0-SNAPSHOT</version>
 </parent>
```

SNAPSHOT Version : Under development

Final



whenever current project ,

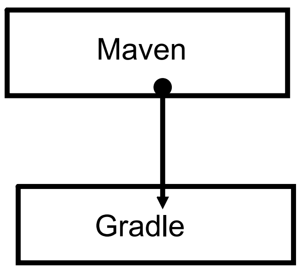
it will always download a new copy

from original src

Project Management Tool

Maven

Gradle



Maven use legacy approach (XML)

Gradle : JAva & Groovy based DSL

Referenced from Maven to overcome certain drawbacks

Maven : Not flexible enough to be customized

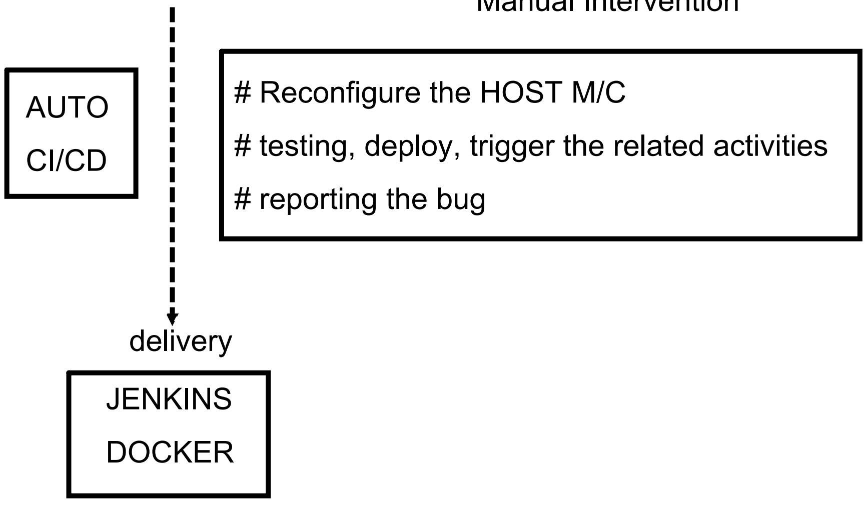
# Platform

# Technology

# IDE

CI/CD : Agile Methodology

#Incremental Artifacts

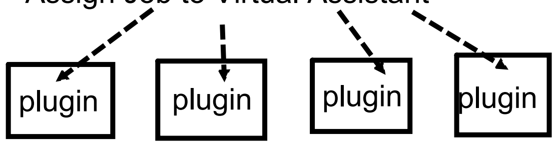


# Reconfigure the HOST M/C
# testing, deploy, trigger the related activities
# reporting the bug

JENKINS : CI/CD

Virtual Assistant

Assign Job to Virtual Assistant



Job : multiple task

CI/CD process in place for spring application created using maven

=> JDK/JRE => Maven => SCM : Git	Jenkins plugins

New incremental artifact ready

1) SCM : GIT (commit)
2) Auto go an fetch new increment from SCM
3) Auto build/package
4) Auto Test
6) Auto Revert back report to team
7) Auto deploy it/containerize it

Install and access it through browser:

Login with credential

We want to automate build process of our spring application

Java
Maven
Git

Need appropriate plugins to interact with these resources

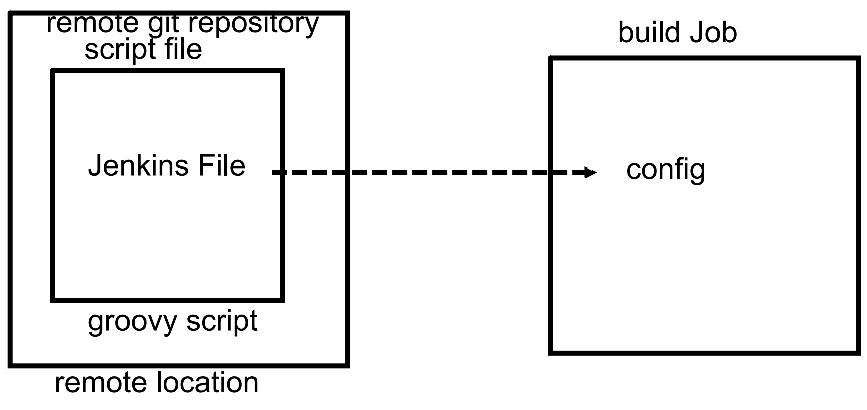
Jenkins will come pre-bundled with plugins for these tool

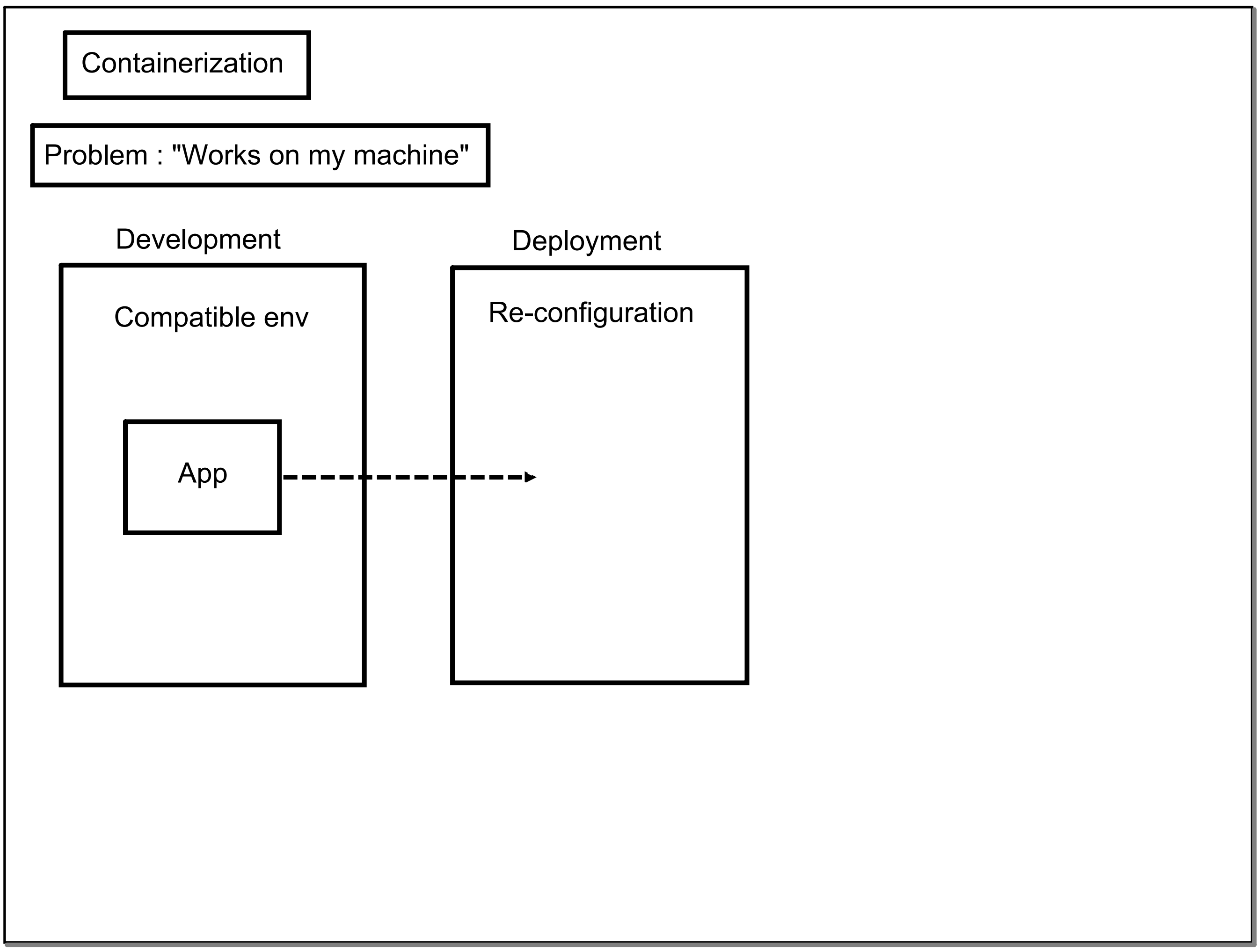
=>Configure the plugin to use these resources

Post-commit

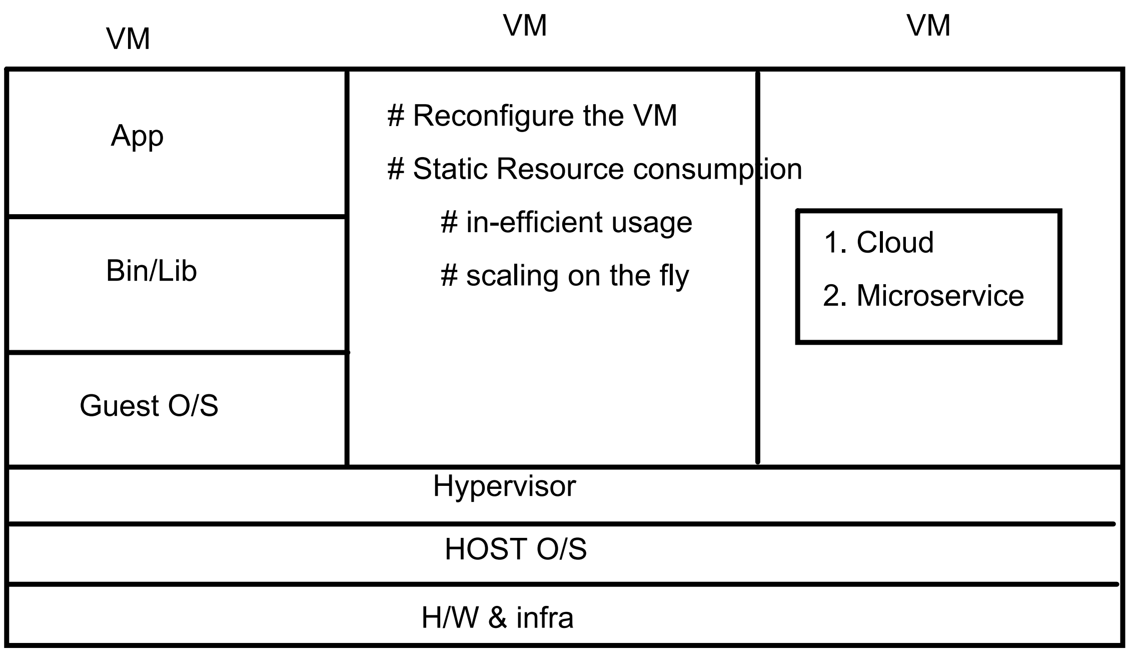
File (plain text file) named as post-commit

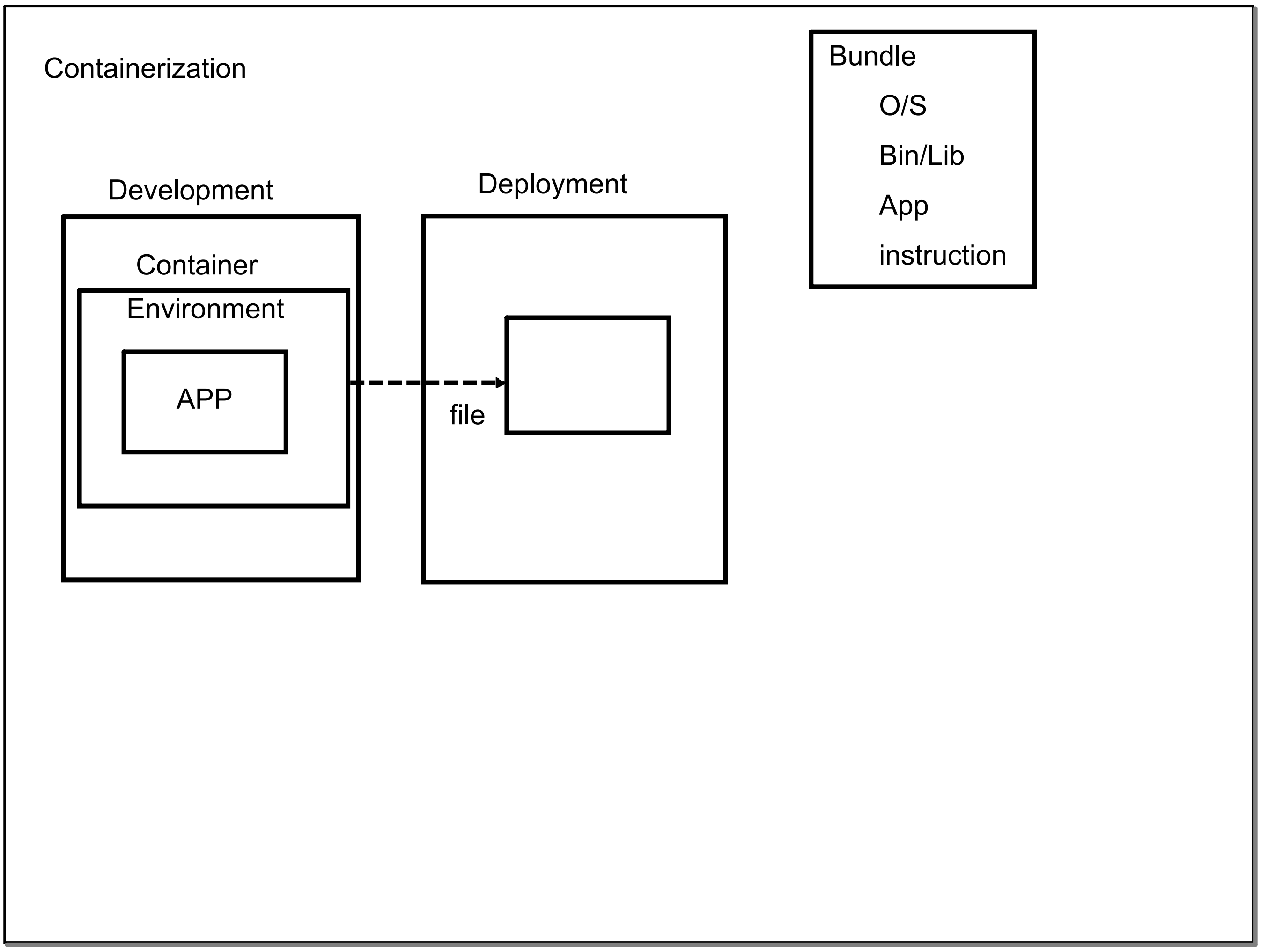
hook notification url of Jenkins



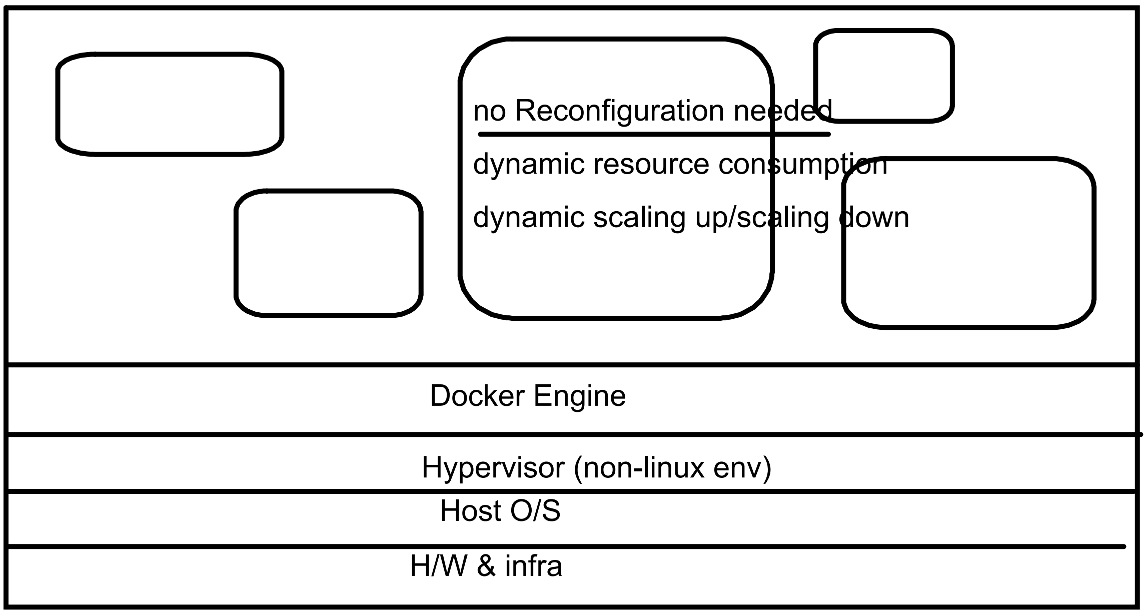

Containerization
Problem : "Works on my machine"
Development
Compatible env
App
Deployment
Re-configuration

Virtual Machine




Containerization
Development
Container
Environment
APP
file
Deployment
Bundle
O/S
Bin/Lib
App
instruction

Containerization


no Reconfiguration needed
dynamic resource consumption
dynamic scaling up/scaling down
Docker Engine
Hypervisor (non-linux env)
Host O/S
H/W & infra

DOCKER TOOL to Containerize

DOCKER

- Client : Development m/c : bundle all require
- Server (Docker Engine)
  Deployment M/C : run the container

Kubernetes

Docker Swarm

Docker : Linux based

Enable Virtualization : System BIOS

Docker Community : Docker Desktop

Image : blueprint of env needed to run the app successfully

Docker Image

Docker Container

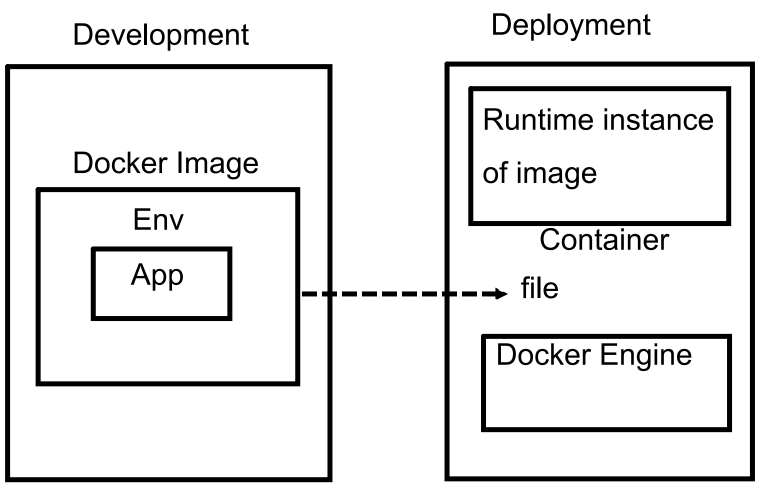


Image :

To prepare VM compatible to app

Spring Boot

Virtual Machine

# install an O/S
# install JDK
# Tomcat Server
# MySQL
# Application jar file
# any config

manifest file

Steps to perform all these activities

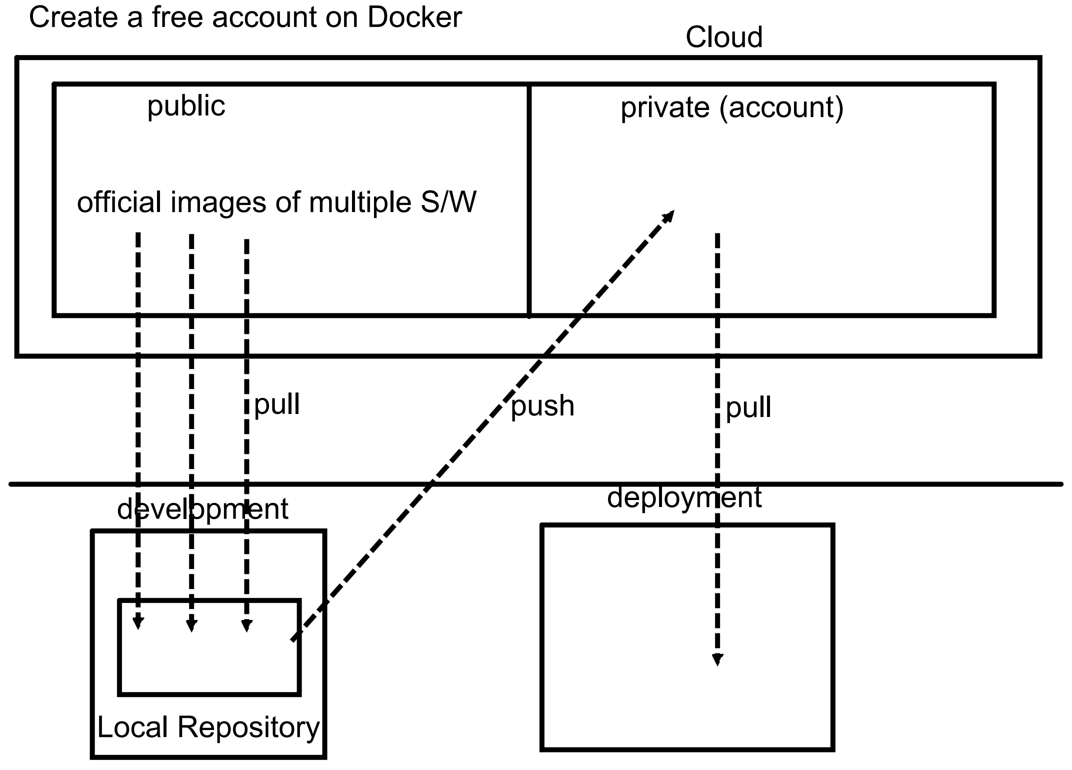
Docker Client : Create a VM based on instruction in manifest
and then create an image(blueprint) of VM
as a file

size = sum of all required software

Create a empty VM, get into MC

Online Cloud Repository of Docker : Docker HUB

Create a free account on Docker



Very Lightweight in nature

Will contain only those binaries or libraries required to run a java application

$\boxed{\text{105MB : Linux O/S installed with JDK 8}}$

To list all docker images in local repository

```
=>docker images
```

To pull docker image from docker hub

```
=> docker pull <image-name>
```

To remove docker image

```
=>docker image rm -f <image-id>
```

To launch/spawn a container on that image

```
=> docker container run <image-name>
      # also pulls from docker hub if not found locally
```

To list all running containers

```
=> docker container ls
```

static web application

# install O/S
# web server (nginx)
# application copied into working dir of nginx server

Virtual Machine

manifest file (Dockerfile)

FROM
LABEL
EXPOSE
WORKDIR
COPY
RUN
CMD

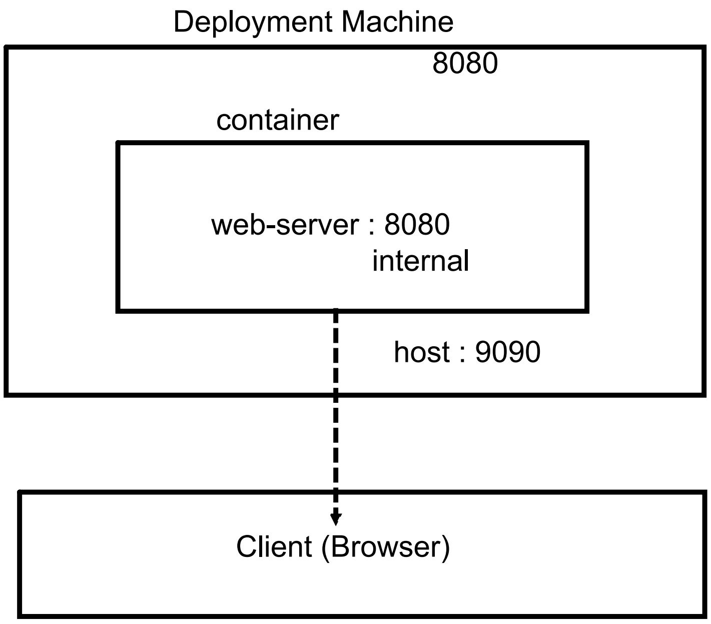
manifest commands

FROM : to install s/w through an images

To Create an image

```
# docker build -t <image-name>:<tag-name> <location of Dockerfile>
```

Web-Server will expose its application on port number :



=> Container port will not conflict with host m/c

=>docker container run -p <host-port>:<internal:port> static-web-app:latest