AGILE : POST - DEVELOPMENT - CI/CD

Maven
JENKINS
DOCKER

1. Manage Dependency
2. Project Structure (std)
3. build
4. Documentation
5. Reporting
6. Distribution

Maven

1. plugin in IDE
2. install the Maven as application in machine
3. Maven batch : mvnw : access maven tool on the fly

GAV Coordinates

- Group ID
- Artifact Id
- Version

Download : official portal

Install

Setting up path variables

- # easy to use on local machine
- # integrate maven with other tool eg: Jenkins

M2_HOME : Home to Maven

M2 : Home to Maven CLI

POM : Project Object Model

- Inbuilt Parent POM file :
  - default config for maven project
- Custom/Local POM
  - custom config for current project

Finally used by maven for project management

- Parent + Custom : Effective POM

Maven CLI

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

Creating a project

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

Maven : plugin based project management tool

```
> mvn <goal>
# it looks for the appropriate plugin : POM (default)
>mvn <plugin>:<goal>
```

Scope of dependency

: when that dependency will be used : visibility of dependency with respect to life cycle phases

build, test, runtime

compile : (default)

  build , test , run

provided

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

  Runtime env will provide

runtime

  test and run

test

  test

system

  ~ provided

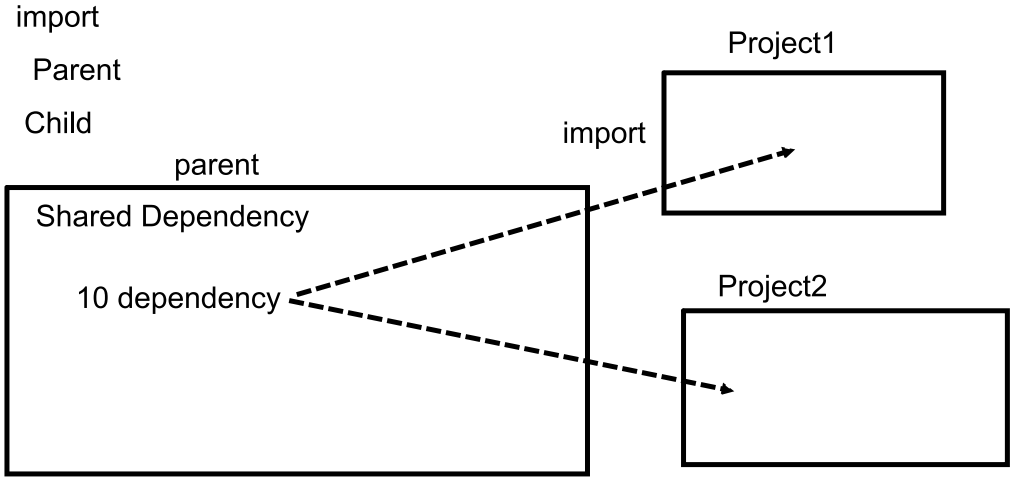
  build, test, run (not to exported : runtime env will also not provide)

  explicit location is required to be mentioned

import

Parent

Child



```
<dependency>
      <groupId>parent group id</groupId>
      <artifactId>.....
      <scope>import</scope>
      <type>pom</type>
</dependency>
```

SNAPSHOT :

.m2 (local repo)

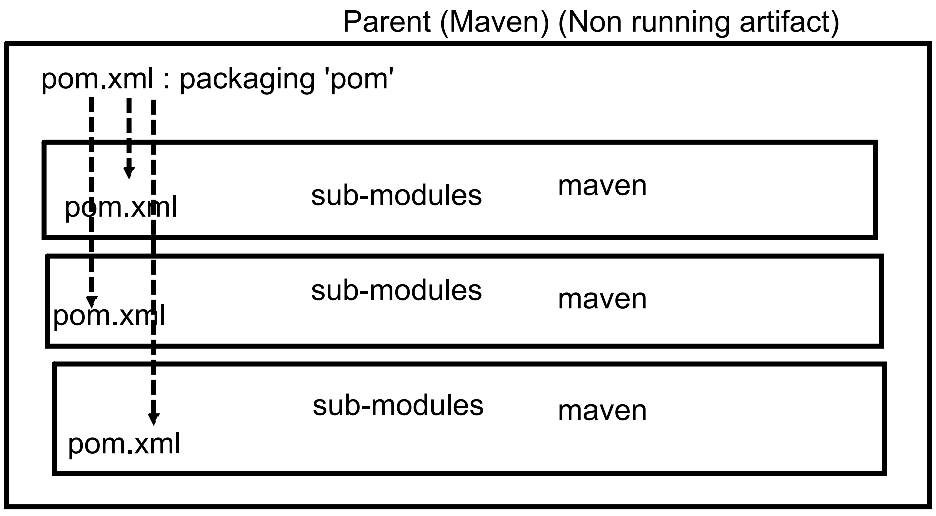
1. look into local repos
2. if not found then go for central

# SNAPSHOT , will always going to be retrieved from the central repo/original source

Multi - Module Project

# Parent + Aggregator

Inheritance + Aggregation



Inheritance : Remove the duplication

Aggregator : any maven goal of task performed on parent will trigger same task in all sub-module

<packaging>pom</packaging>

Makes parent pom.xml

No more a running artifact

Child modules to be created under parent project folder

Aggregation

```
  <modules>
       <module>child-module1</module>
       <module>child-module2</module>
       <module>child-module3</module>
</modules>
```

1. Any task on parent will trigger same on these child modules
2. defines the order of maven goals

Inheritance

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

Maven : Project Management Tool

Gradle :

$$\begin{tikzcd}\boxed{\text{Maven}} \arrow[d, dashed] \\ \boxed{\text{Gradle}}\end{tikzcd}$$

Refrenced from Maven to overcome drawbacks of Maven

Maven : Not flexible enough to be customized

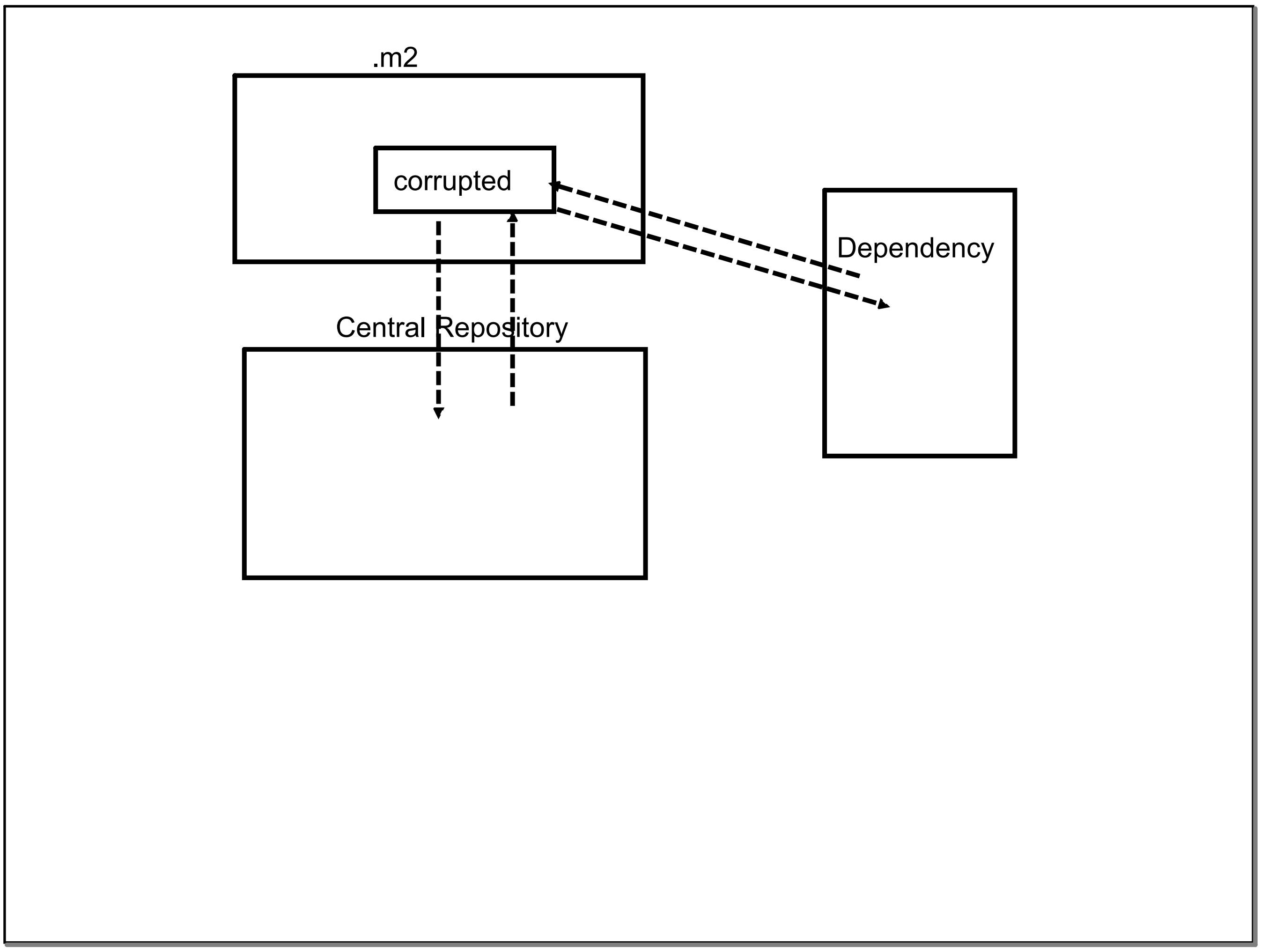
# Platform

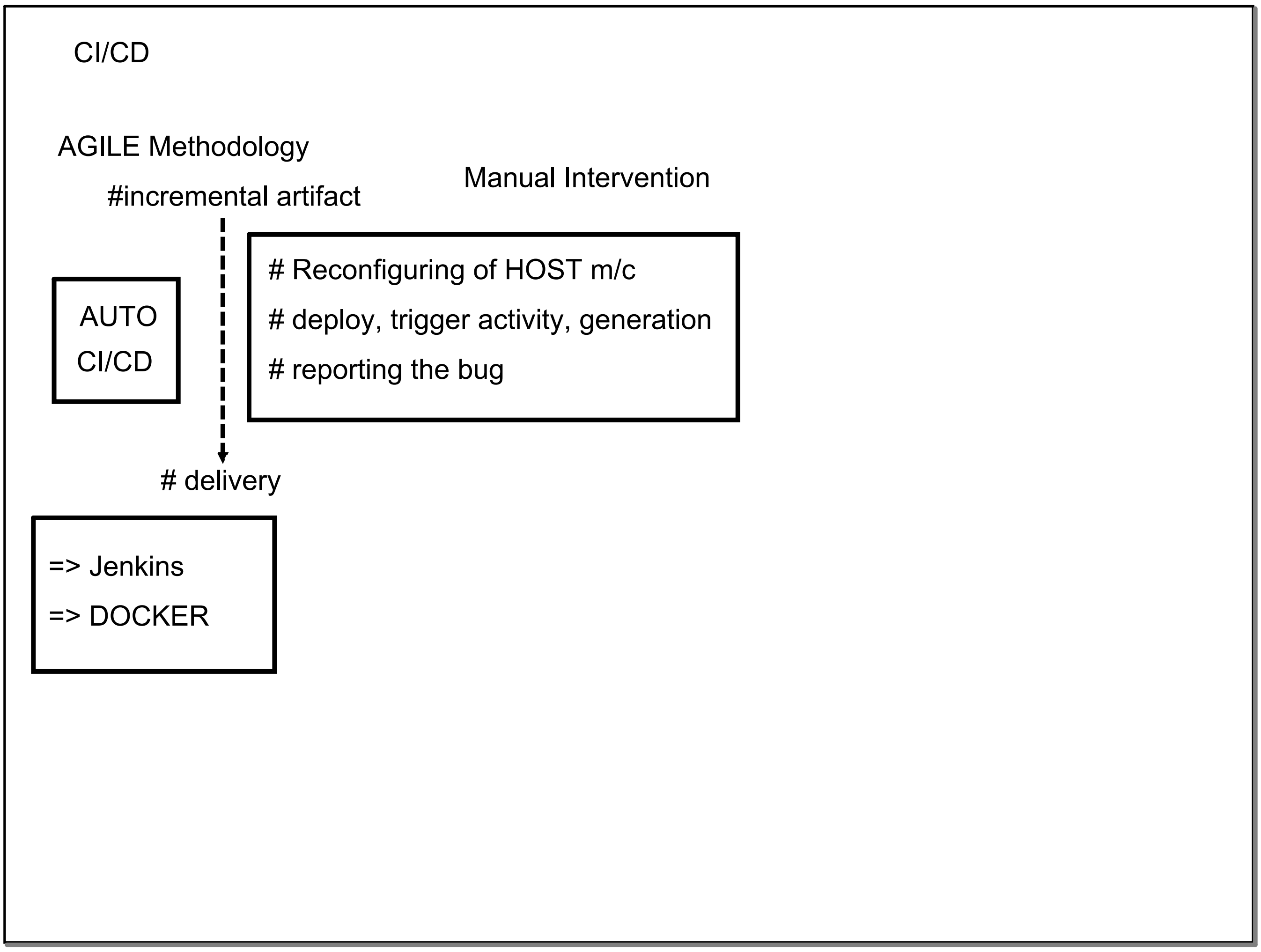
# Technology

# IDE

Maven : pom ---- XML (Legacy)

Gradle : DSL like groovy


.m2
corrupted
Dependency
Central Repository



CI/CD

AGILE Methodology

#incremental artifact

Manual Intervention

AUTO
CI/CD

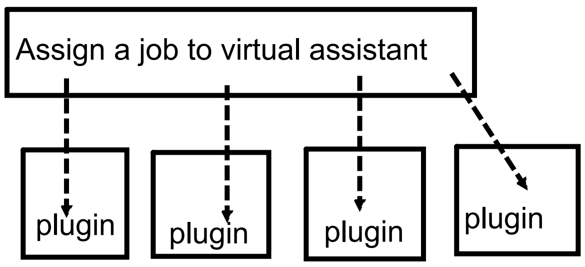
# Reconfiguring of HOST m/c
# deploy, trigger activity, generation
# reporting the bug

# delivery

=> Jenkins
=> DOCKER

JENKINS : CI/CD

Virtual Assistant



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

# JDK/JRE
# Maven
# SCM : GIT

plugin

> mvn clean package

Automated by Jenkins

1. Initiate a git repository of this project

Git hook

.git (hidden)

hooks

hook file (git phase)

post-commit

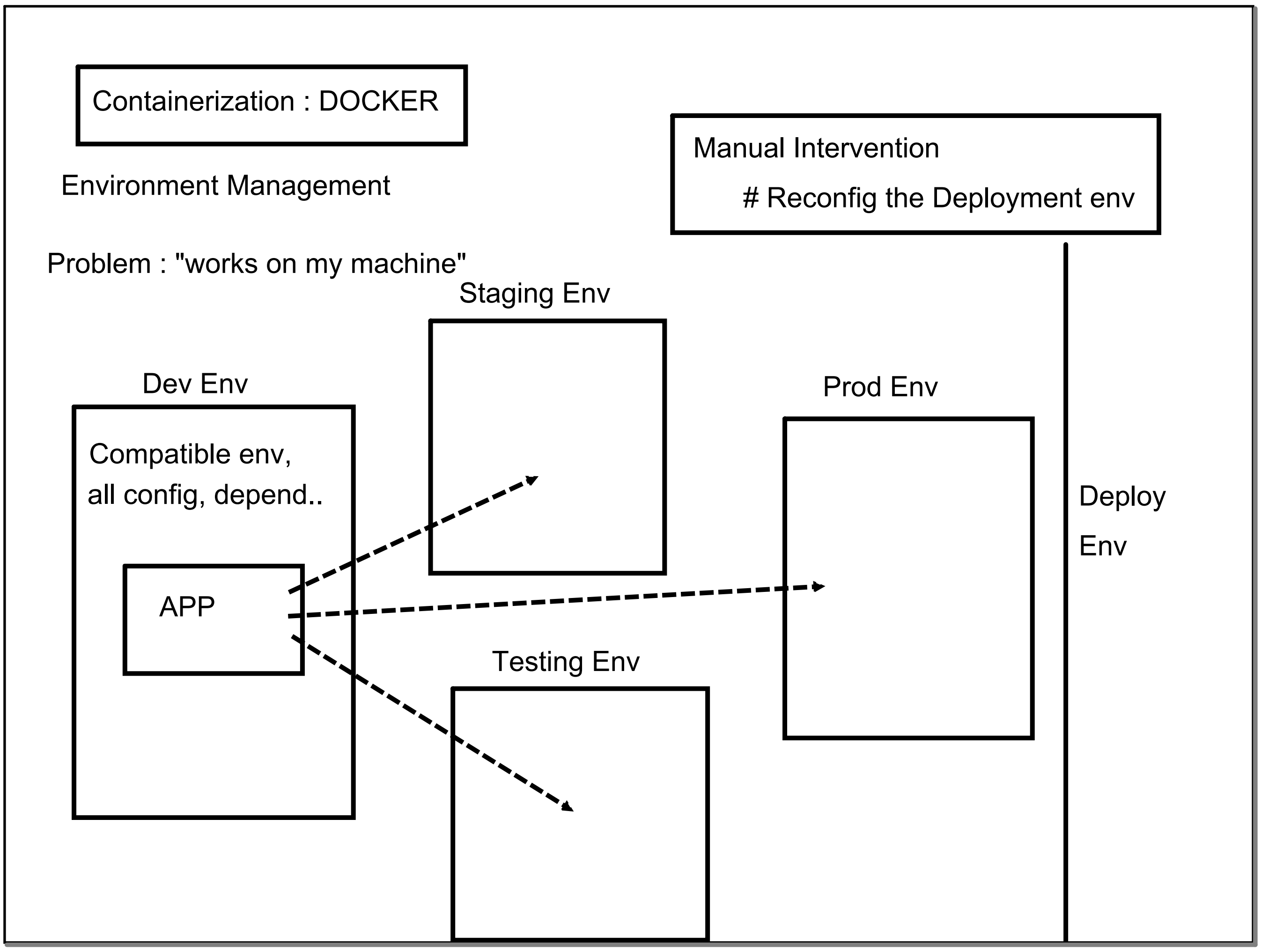
Automated Email system, need to have

Email Extenstion plugin

# Configure the plugin

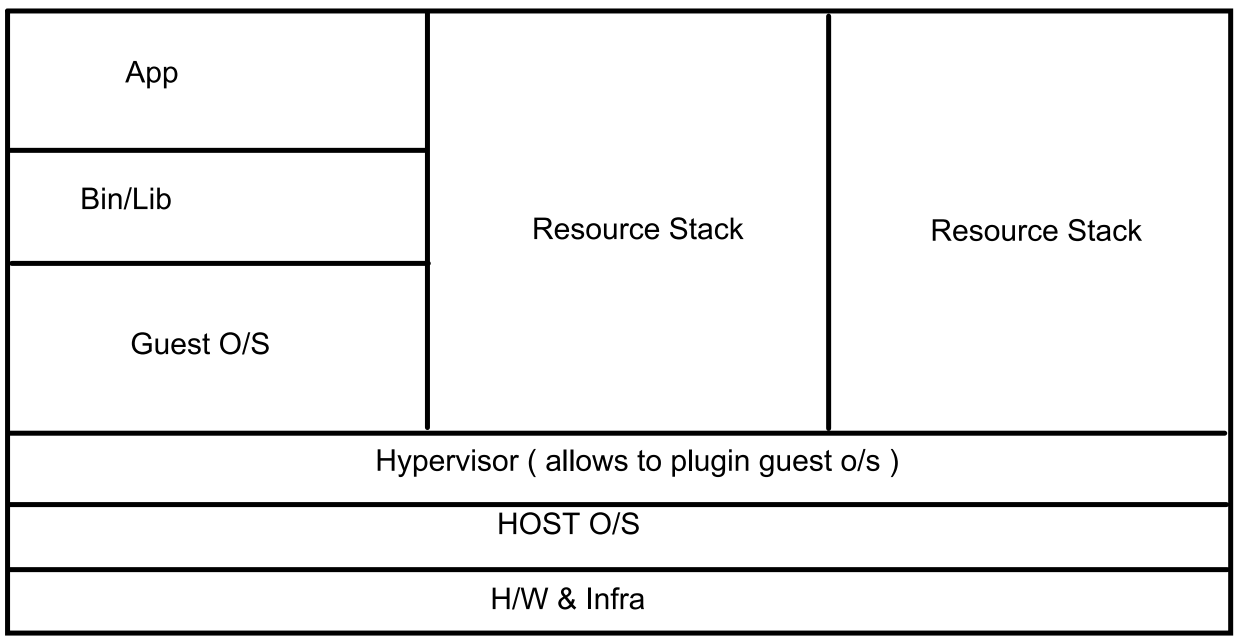
DSL : groovy

script for build job, placed remotely, configured in JEnkins to read through git

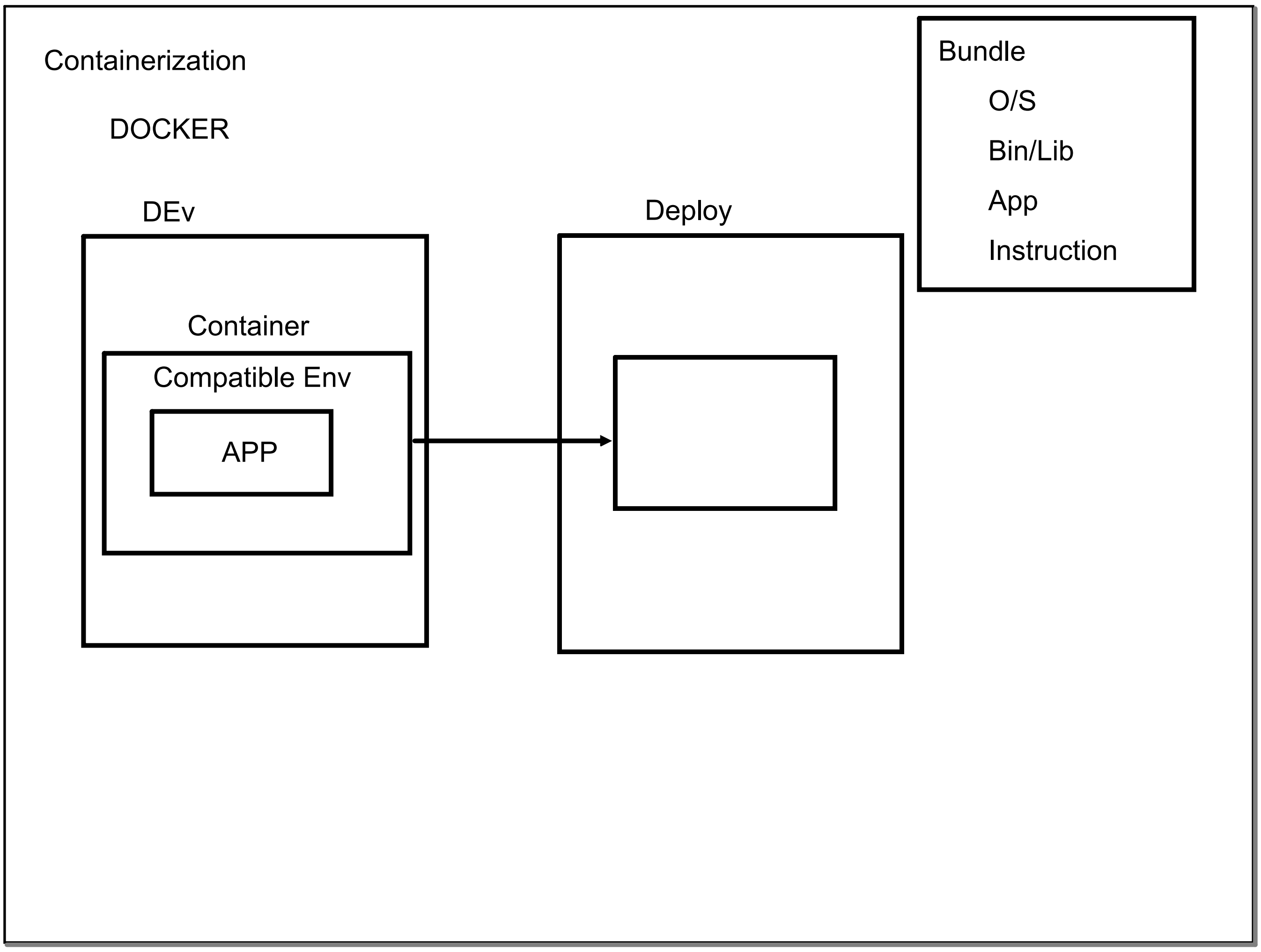

Containerization : DOCKER
Environment Management
Problem : "works on my machine"
Manual Intervention
# Reconfig the Deployment env
Staging Env
Dev Env
Compatible env,
all config, depend..
APP
Prod Env
Testing Env
Deploy
Env

Virtual Machine

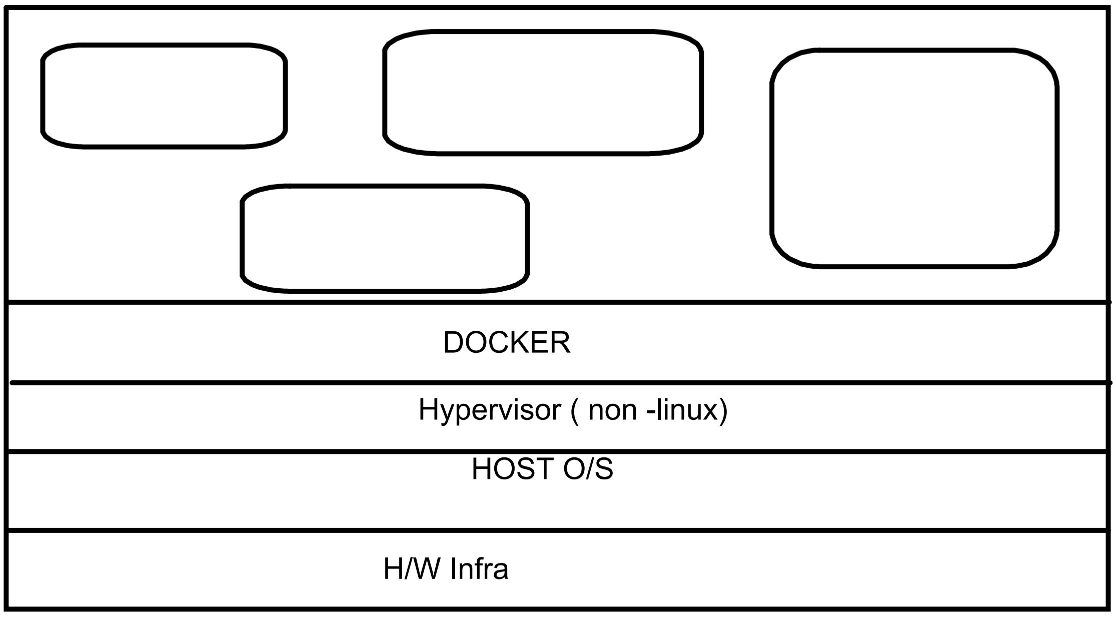
Resource Stack



1. Reconfiguration is required
2. Static Resource Stack
    # Keep on occupying Fixed amount of resource
    # Scaling is not possible


Containerization
DOCKER
DEv
Container
Compatible Env
APP
Deploy
Bundle
O/S
Bin/Lib
App
Instruction

Container Env



# No Reconfiguration is required

# No Static Consumption of resources

  # use the system resources on the fly

  # scale up/down can be down

$$\begin{tikzcd}
 & \text{Client} \\
\boxed{\text{DOCKER}} \arrow[ru] \arrow[rd] & \\
 & \text{Server}
\end{tikzcd}$$

Client: Dev : Create the env (container)

Server: Deploy : Run the application as container

Community Edition (Client + Server)

Docker Linux Based (Win 10)

1 : Enable Virtualization (System BIOS)

Docker Desktop

Docker Image

Docker Container

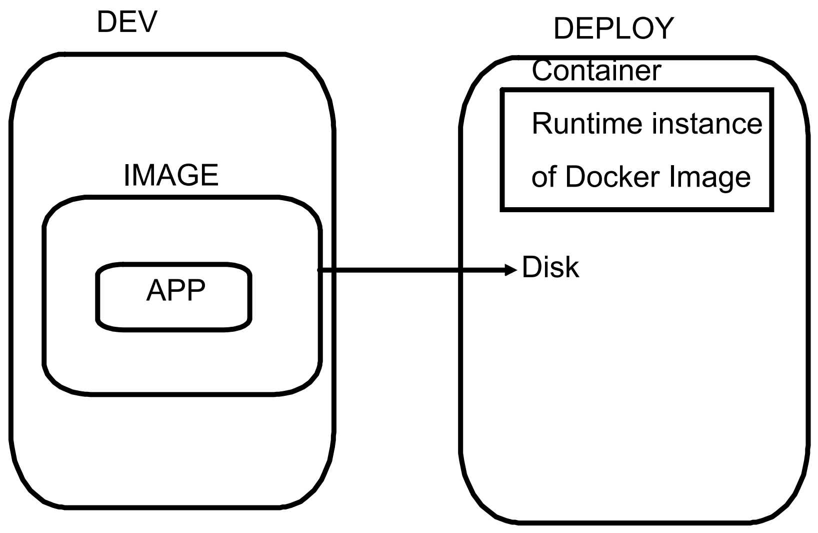


IMAGE
- Blueprint
  - Bundle of O/S
  - JDK
  - instruction to run

instruction to download

(not recommended)

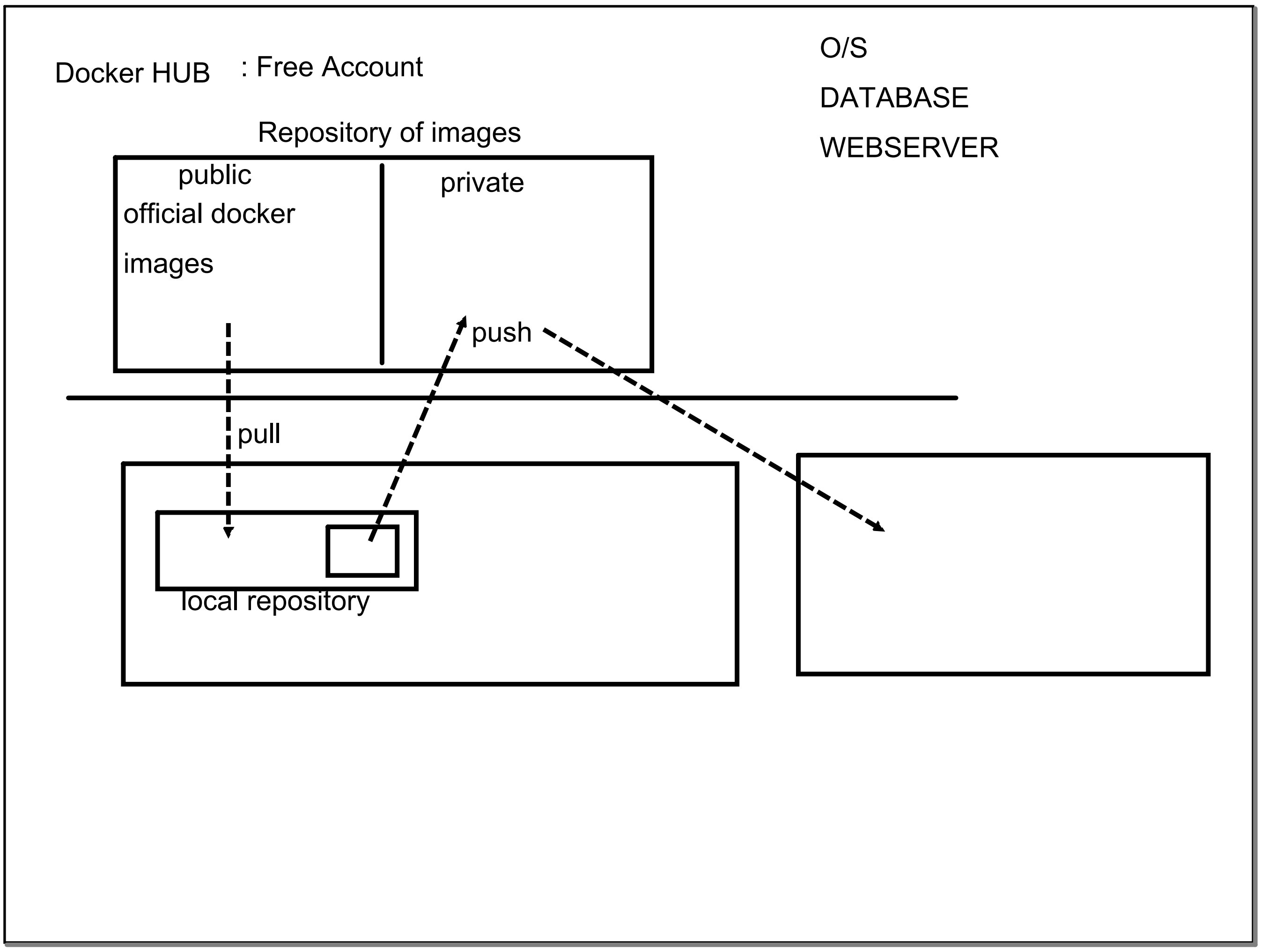

Docker HUB
: Free Account
O/S
DATABASE
WEBSERVER
Repository of images
public
official docker
images
private
push
pull
local repository

IMAGE
- Blueprint
  - Bundle of O/S
  - JDK
  - instruction to run

instruction to download
(not recommended)

O/S : Bare minimum binaries to support application

> Listing all images : docker images

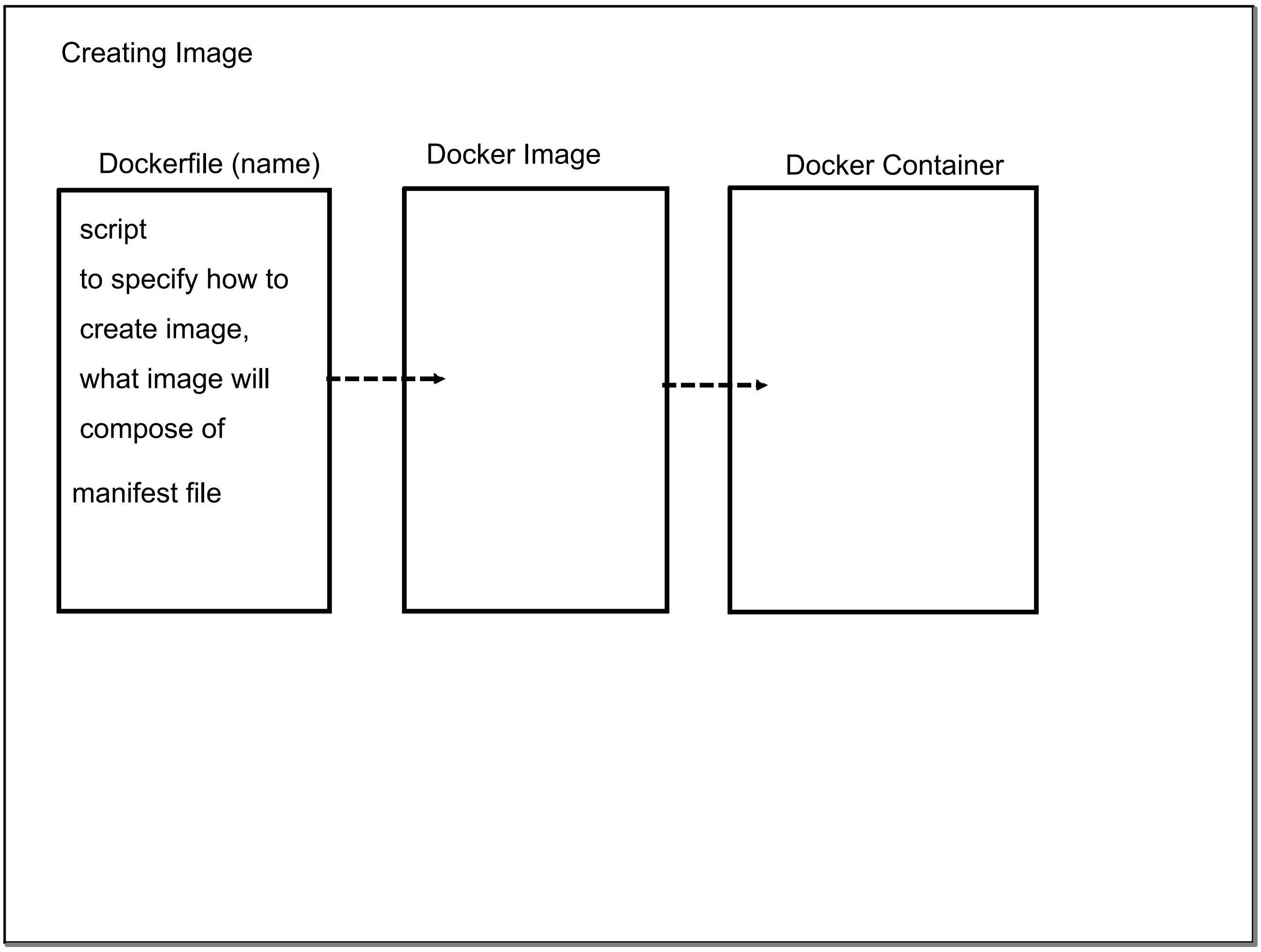
> Pull an image : docker pull <image name>:[tag/version]

by default most recent version of that image

> to run a container : docker container run <image-name>

1. Local Repository
2. Pull it from docker hub

> List Running Container : docker container ls


Creating Image
Dockerfile (name)
script
to specify how to
create image,
what image will
compose of
manifest file
Docker Image
Docker Container

Scripts Command : Instructions to prepare a virtual machine

=> FROM

FROM <image-name>

- install O/S
- install JDK
- install WEB SERVER
- install MySQL

# install software on virtual machine

=> RUN

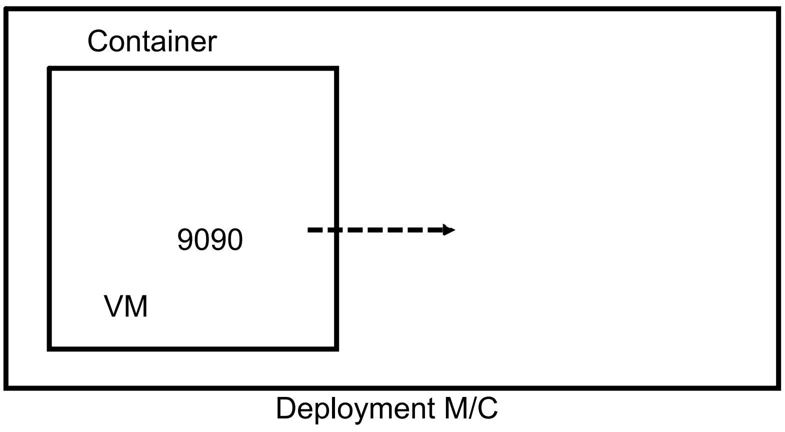
RUN mkdir app

- path setting
- downloading updates
- cleaning cache
- configuration
- new folder/file

==>COPY/ADD

Copy the resources/files from dev m/c to VM

==>EXPOSE port



Creating Docker Image ~ Dockerfile

Static Web Application : 1 HTML file

Dockerfile

- Install O/S
- install Web Server
- Copy our Application Code
- Deploy it on Web Server
- Write instructions to run web server
- expose the port on which server is up

nginx: deploy a static web application
place the static res into:
/usr/share/nginx/html
(default application of nginx)

COPY <src> <dest>

src : file system dev m/c
dest : file system of VM

Creating an image:

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

Launching container by mapping port number

```
.>docker container run -p <host port number>: <internal port number> <image-name>
```

Spring boot RestBased Web Application

install O/S

install JDK

application (packaged : jar)

instruction to run the jar file (container)

Build Job for Jenkins

1. Checkout src code from GIT
2. build the project
3. Run the test case
4. if possible revert with email
5. package the project
6. Dockerize the application

$$\begin{tikzcd} \text{Jenkins} & + & \text{Docker (IP)} \\ & \text{Docker image : plumbing agent} \arrow[u, dashed] & \end{tikzcd}$$

Configure the Jenkins for Docker

1. Install Docker plugin to Jenkins
2. Global Tool Config (local instance)
3. Configure System

Docker Name

Docker Host URI

Testing the connection

Enable it

Configure Docker Agent

    # Label

    # Enable

    # Docker Reference Name

    # Agent Image name : benhall/dind-jenkins-agent:v2

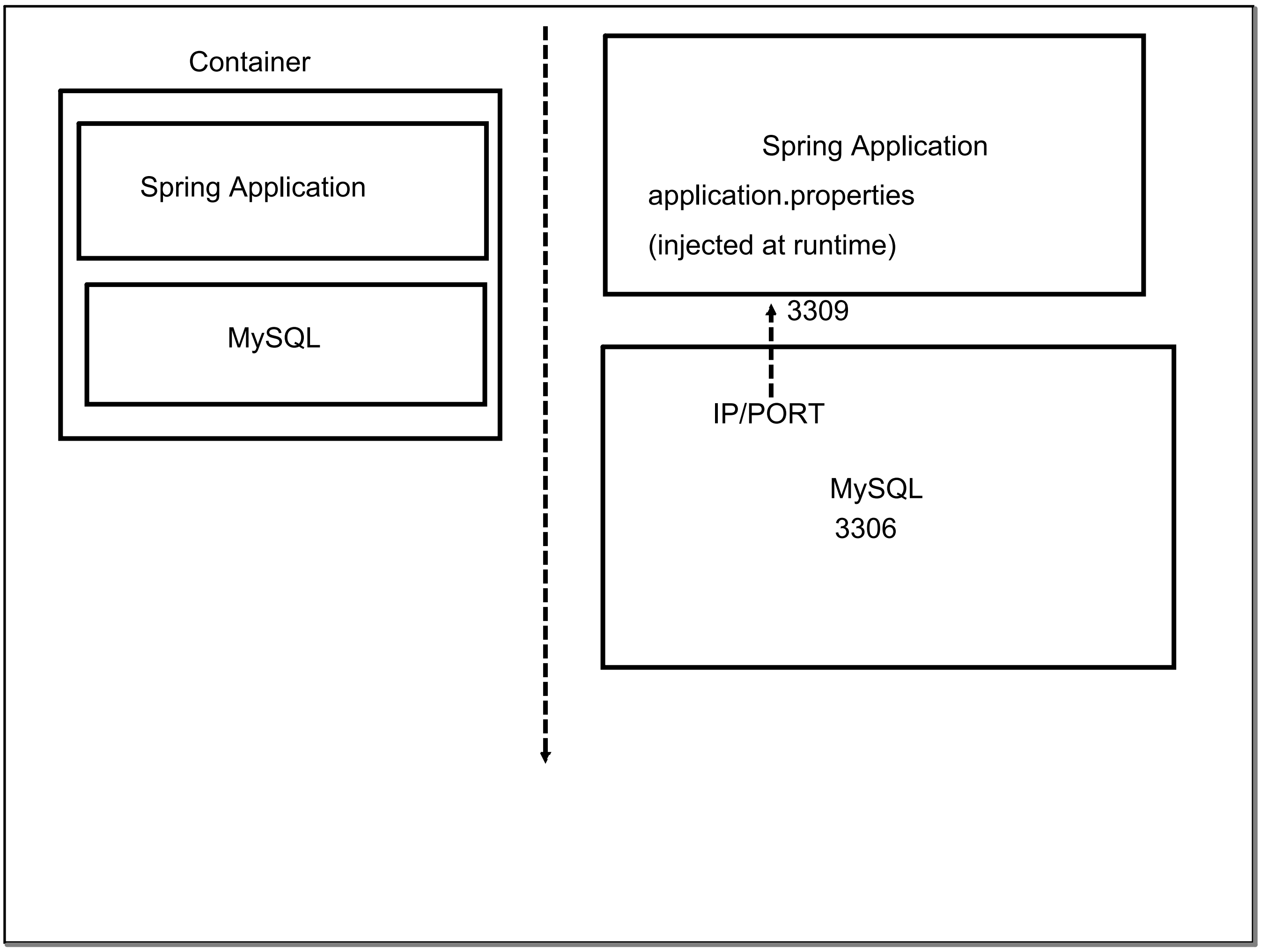
    # Connect Method : Connect with SSH

Container Settings

    Volumes : /var/run/docker.sock:/var/run/docker.sock

Maven plugin to docker the spring application

1. Add plugin in pom.xml
2. Configure plugin for image creation
3. Will create a docker image auto when build is done


Container
Spring Application
MySQL
Spring Application
application.properties
(injected at runtime)
3309
IP/PORT
MySQL
3306

application.properties

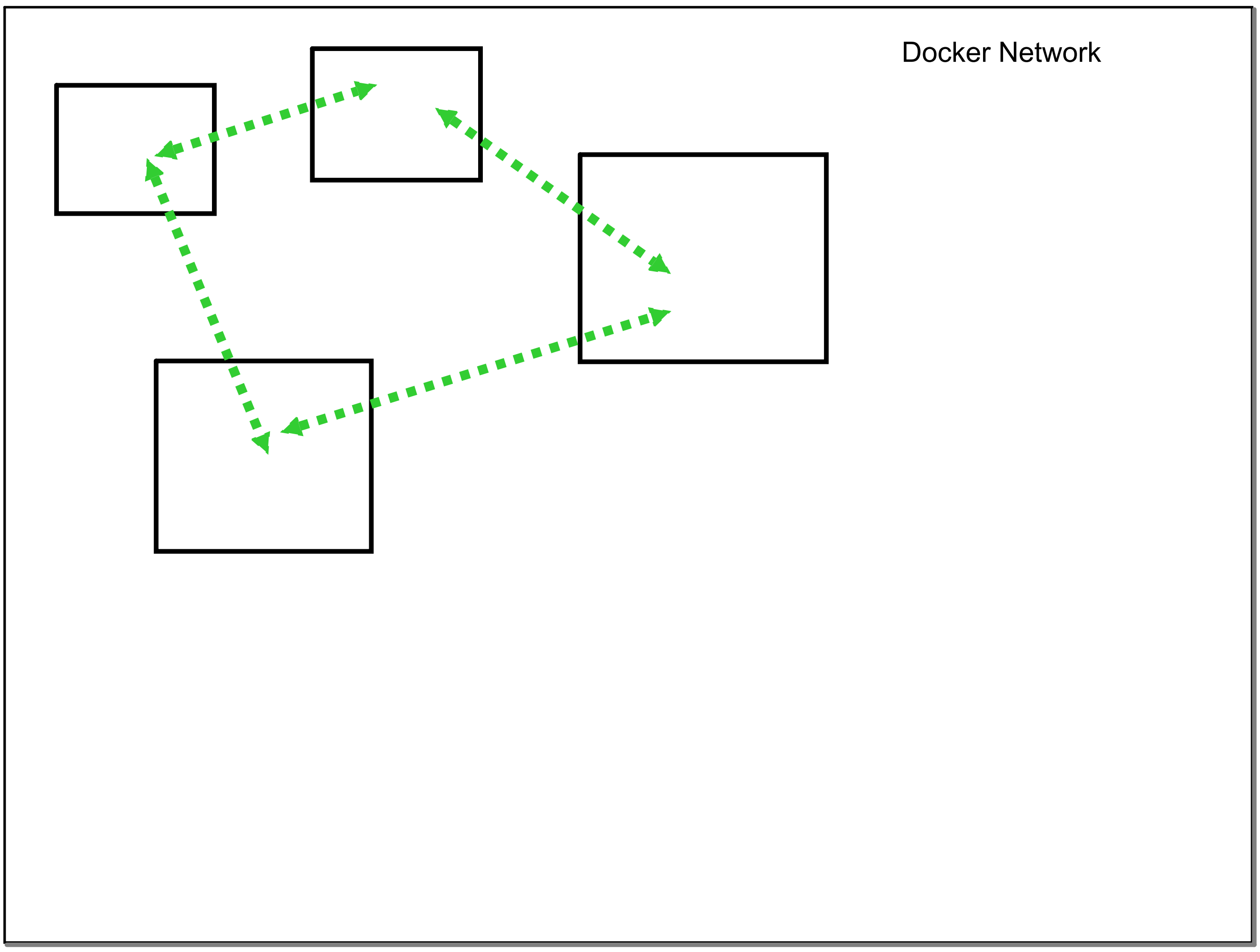
```
spring.datasource.url = jdbc:mysql://localhost:3306/user_db
==> spring.datasource.url = jdbc:mysql://${INJ_HOSTNAME:localhost}:${INJ_PORT :3306}/$
{INJ_DB:user_db}
==>spring.datasource.username = ${INJ_USERNAME:root}
------------------------
```

#Launch Mysql

```
docker container run
-p 3307:3306
--env MYSQL_ROOT_PASSWORD=<root-pass>
--env MYSQL_DATABASE = <db name>
--name <mysql custom name>
<mysql-image>
```

Spring App

```
docker container run
-p 9093:9090
--link = <mysql custom name>
--env INJ_HOSTNAME = localhost
--env INJ_PORT = 3307
--env INJ_USERNAME = root
--env INJ_PASSWORD = root
--env INJ_DB = dbname
<app image name>
```


Docker Network