Module 3: Provisioning Docker Images

- Introducing the Dockerfile
- Creating a Dockerfile
- Building images manually
- Storing and retrieving Docker Images from Docker Hub
- Building images using Continuous Integration tools
- Inspecting a Dockerfile from DockerHub
- Lab Exercises

Introducing the Dockerfile

A `Dockerfile` is a text document that contains all the commands a user could call on the command line to assemble an image.

Example

$ cat /home/user/my-docker/Dockerfile

```
FROM python:2.7-slim
WORKDIR /app
ADD app.py /app
ADD requirements.txt /app
RUN pip install --trusted-host pypi.python.org -r requirements.txt
EXPOSE 80
ENV name world
CMD ["python","app.py"]
```

Usage

```
$ docker build .
Sending build context to Docker daemon  6.51 MB
...
```

Introducing the Dockerfile

- ENV - to set environment variables
- EXPOSE - to expose ports
- FROM - base image
- LABEL - to add metadata to image
- HEALTHCHECK - to check if container is running
- USER - to set user and group
- VOLUME - to specify mount point from external host
- WORKDIR - workdir to run any of the commands

Introducing the Dockerfile

- ARG - variable used during build time
- CMD - to provide defaults to executing container
- RUN - to execute commands in new layer
- COPY - Copy file,dir or remote url to image
- ADD - Copy file,dir or remote url to image
- ENTRYPOINT - to configure container as executable
- MAINTAINER - the image maintainer

RUN COPY ADD instructions create new layers in the image stack - refer layering section

Creating Dockerfile (s)

```
FROM bitnami/minideb-extras:jessie-r23
LABEL maintainer "Bitnami <containers@bitnami.com>"

# Install required system packages and dependencies
RUN install_packages libapr1 libaprutil1 libc6 libexpat1 libffi6 libgmp10 libgnutls-deb0-28 libhogweed2 libldap-2.4-2 libnettle4
libp11-kit0 libpcre3 libsasl2-2 libssl1.0.0 libtasn1-6 libuuid1 zlib1g
RUN bitnami-pkg unpack apache-2.4.29-1 --checksum
42114e87aafb1d519ab33451b6836873bca125d78ce7423c5f7f1de4a7198596
RUN ln -sf /opt/bitnami/apache/htdocs /app

COPY rootfs /

ENV APACHE_HTTPS_PORT_NUMBER="443" \
    APACHE_HTTP_PORT_NUMBER="80" \
    BITNAMI_APP_NAME="apache" \
    BITNAMI_IMAGE_VERSION="2.4.29-r1" \
    PATH="/opt/bitnami/apache/bin:$PATH"

EXPOSE 80 443

WORKDIR /app
ENTRYPOINT ["/app-entrypoint.sh"]
CMD ["nami","start","--foreground","apache"]
```

Dockerfile - Example

```
FROM jenkinsci/jenkins:latest
LABEL maintainer "r1co@post-box.cc"

USER root

# install docker cli
RUN mkdir -p /tmp/_install && cd /tmp/_install && wget https://get.docker.com/builds/Linux/x86_64/docker-latest.tgz  && tar
-xvzf docker-latest.tgz && cd docker && cp docker /usr/bin/docker && rm -rf  /tmp/_install
RUN chmod +x /usr/bin/docker
# add jenkins to docker group
RUN groupadd -g 999 docker
RUN usermod -a -G docker jenkins
# install docker-compose
RUN curl -L https://github.com/docker/compose/releases/download/1.7.1/docker-compose-`uname -s`-`uname -m` >
/usr/local/bin/docker-compose
RUN chmod +x /usr/local/bin/docker-compose
USER jenkins
```

Build Image manually

Build an image from a Dockerfile

Usage

```
docker build [OPTIONS] PATH | URL | -
```

Options

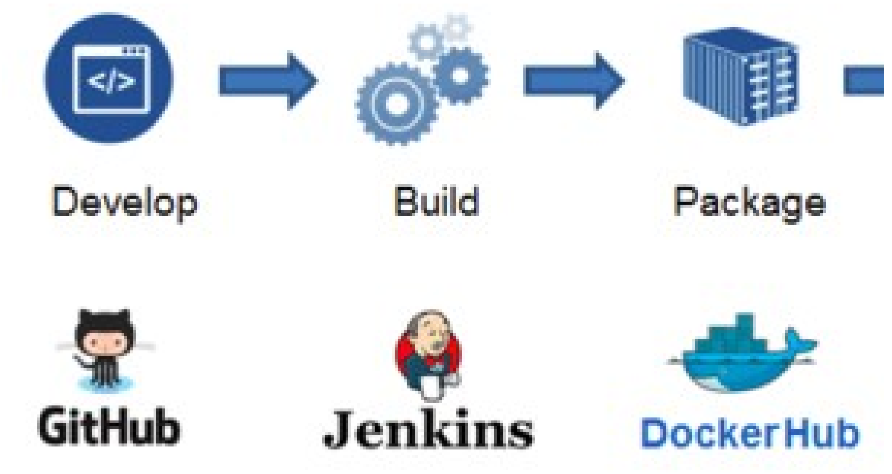
```
--build-arg set build variables
--compress compress the build context using gzip
--file name of the Dockerfile
--label set metadata for image
--rm remove intermediate containers post build
--tag name and optionally tag in the name:tag format
--ulimit options
...
```

Docker Hub - store & retrieve

https://hub.docker.com (register and create login)

- docker tag alpine rajeshgheware/alpine:rajesh
- docker push rajeshgheware/alpine:rajesh
- docker pull rajeshgheware/alpine:rajesh

Build Image using CI / Jenkins



docker run -p 8080:8080 -p 50000:50000 -v /var/run/docker.sock:/var/run/docker.sock -d r1co/jenkins-docker

Dockerfile - Docker Hub

https://hub.docker.com/u/bitnami/

https://hub.docker.com/u/springio/

https://hub.docker.com/r/sebp/elk/~/dockerfile/

Lab Exercises

Node JS App

- Create simple nodejs app to print caller address and node hostname
- Create Dockerfile by tagging the image to match your docker ID
- Run the container & verify that app is working
- Push the image to your docker hub repo having image name and tag properly
- Build this nodejs app using containerized CI - Jenkins (r1co/jenkins-docker, Use jenkins file)
- Verify CI deploys docker images to you docker hub repo
- Modify nodejs app treating the change as version 2 changes
- Verify that CI picks up the change creates next version of docker image and deploys to docker hub
- Tag the code as v3.0 and push the tag to github and observe the docker hub repo for the image corresponding to this tag

SSH Server

- Create Dockerfile to build ssh server image based on Ubuntu