Group A Assignment No. 1(c)

Aim:-Docker installation in Ubuntu operating system.

Objectives:-To study Docker and its installation. To understand its benefits over vmware or virtual box

Theory:-

Docker allows you to package an application with all of its dependencies into a standardized unit for software development.Docker containers wrap up a piece of software in a complete file system that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in.

Properties of Docker:-

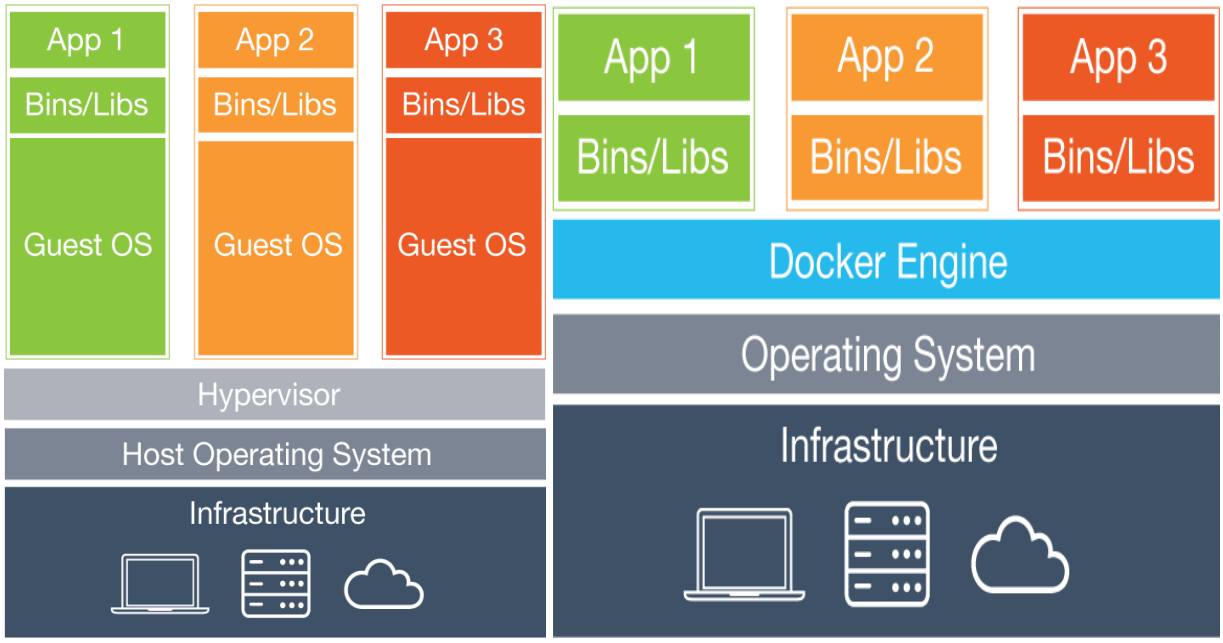
Lightweight : Containers running on a single machine all share the same operating system kernel so they start instantly and make more efficient use of RAM. Images are constructed from layered file systems so they can share common files, making disk usage and image downloads much more efficient.

Open : Docker containers are based on open standards allowing containers to run on all major Linux distributions and Microsoft operating systems with support for every infrastructure.

Secure : Containers isolate applications from each other and the underlying infrastructure while providing an added layer of protection for the application.

How is this different from virtual machines?

Containers have similar resource isolation and allocation benefits as virtual machines but a different architectural approach allows them to be much more portable and efficient.



Virtual Machines : Each virtual machine includes the application, the necessary binaries and libraries and an entire guest operating system - all of which may be tens of GBs in size.

Containers : Containers include the application and all of its dependencies, but share the kernel with other containers. They run as an isolated process in userspace on the host operating system. They're also not tied to any specific infrastructure – Docker containers run on any computer, on any infrastructure and in any cloud.

Helps you build better software : When your app is in Docker containers, you don't have to worry about setting up and maintaining different environments or different tooling for each language. Focus on creating new features, fixing issues and shipping software.

Accelerate Developer Onboarding : Stop wasting hours trying to setup developer environments, spin up new instances and make copies of production code to run locally. With Docker, you can easily take copies of your live environment and run on any new endpoint running Docker.

Empower Developer Creativity : The isolation capabilities of Docker containers free developers from the worries of using "approved" language stacks and tooling. Developers can

use the best language and tools for their application service without worrying about causing conflict issues.

Eliminate Environment Inconsistencies : By packaging up the application with its configs and dependencies together and shipping as a container, the application will always work as designed locally, on another machine, in test or production. No more worries about having to install the same configs into a different environment.

Easily Share and Collaborate on Applications : Docker creates a common framework for developers and sysadmins to work together on distributed applications

Distribute and share content : Store, distribute and manage your Docker images in your Docker Hub with your team. Image updates, changes and history are automatically shared across your organization.

Simply share your application with others : Ship one or many containers to others or downstream service teams without worrying about different environment dependencies creating issues with your application. Other teams can easily link to or test against your app without having to learn or worry about how it works.

Ship More Software Faster : Docker allows you to dynamically change your application like never before from adding new capabilities, scaling out services to quickly changing problem areas.

Quickly Scale : Docker containers spin up and down in seconds making it easy to scale an application service at any time to satisfy peak customer demand, then just as easily spin down those containers to only use the resources you need when you need it

Easily Remediate Issues : Docker make it easy to identify issues and isolate the problem container, quickly roll back to make the necessary changes then push the updated container into production. The isolation between containers make these changes less disruptive than traditional software models.

Prerequisites for Docker installatio:

Docker requires a 64-bit installation regardless of your Ubuntu version. Additionally, your kernel must be 3.10 at minimum. The latest 3.10 minor version or a newer maintained version are also acceptable. Kernels older than 3.10 lack some of the features required to run Docker containers. These older versions are known to have bugs which cause data loss and frequently panic under certain conditions. To check your current kernel version, open a terminal and use uname -r to display your kernel version:

$ uname -r
3.11.0-15-generic

Caution Some Ubuntu OS versions **require a version higher than 3.10** to run Docker, see the prerequisites on this page that apply to your Ubuntu version.

For Precise 12.04 (LTS)

For Ubuntu Precise, Docker requires the 3.13 kernel version. If your kernel version is older than 3.13, you must upgrade it. Refer to this table to see which packages are required for your environment:

linux-image-generic-lts-trusty	Generic Linux kernel image. This kernel has AUFS built in. This is required to run Docker.

To upgrade your kernel and install the additional packages, do the following:

1. Open a terminal on your Ubuntu host.
2. Update your package manager.
3. $ sudo apt-get update
4. Install both the required and optional packages.
5. $ sudo apt-get install linux-image-generic-lts-trusty

   Depending on your environment, you may install more as described in the preceding table.

6. Reboot your host.
7. $ sudo reboot
8. After your system reboots, go ahead and install Docker.

For Saucy 13.10 (64 bit)

Docker uses AUFS as the default storage backend. If you don't have this prerequisite installed, Docker's installation process adds it.

Installation Steps

Make sure you have installed the prerequisites for your Ubuntu version. Then, install Docker using the following:

1. Log into your Ubuntu installation as a user with sudo privileges.
2. Verify that you have curl installed.
3. $ which curl

   If curl isn't installed, install it after updating your manager:

   $ sudo apt-getupdate
   $ sudo apt-get install curl

4. Get the latest Docker package.
5. $ curl -sSL https://get.docker.com/ | sh

   The system prompts you for your sudo password. Then, it downloads and installs Docker and its dependencies.

Note: If your company is behind a filtering proxy, you may find that the apt-key command fails for the Docker repo during installation. To work around this, add the key directly using the following:

$ curl -sSL https://get.docker.com/gpg | sudo apt-key add -

1. Verify docker is installed correctly.
2. $ sudodockerrun hello-world

   This command downloads a test image and runs it in a container.

Optional configurations for Docker on Ubuntu

This section contains optional procedures for configuring your Ubuntu to work better with Docker.

- Create a docker group
- Adjust memory and swap accounting
- Configure Docker to start on boot

Create a Docker group

The docker daemon binds to a Unix socket instead of a TCP port. By default that Unix socket is owned by the user root and other users can access it with sudo. For this reason, docker daemon always runs as the root user.

To avoid having to use sudo when you use the docker command, create a Unix group called docker and add users to it. When the docker daemon starts, it makes the ownership of the Unix socket read/writable by the docker group.

Warning: The docker group is equivalent to the root user; For details on how this impacts security in your system, see *Docker Daemon Attack Surface* for details.

To create the docker group and add your user:

1. Log into Ubuntu as a user with sudo privileges.

   This procedure assumes you log in as the ubuntu user.

2. Create the docker group and add your user.
3. $ sudousermod -aGdockerubuntu

4. Log out and log back in.

   This ensures your user is running with the correct permissions.

5. Verify your work by running docker without sudo.
6. $ dockerrun hello-world

   If this fails with a message similar to this:

   Cannot connect tothe Docker daemon. Is 'docker daemon' running onthishost?

   Check that the DOCKER_HOST environment variable is not set for your shell. If it is, unset it.

Adjust memory and swap accounting

When users run Docker, they may see these messages when working with an image:

WARNING: Your kernel does not support cgroup swap limit. WARNING: Your
kernel does not support swap limit capabilities. Limitation discarded.

To prevent these messages, enable memory and swap accounting on your system. Enabling memory and swap accounting does induce both a memory overhead and a performance degradation even when Docker is not in use. The memory overhead is about 1% of the total available memory. The performance degradation is roughly 10%.

To enable memory and swap on system using GNU GRUB (GNU GRand Unified Bootloader), do the following:

1. Log into Ubuntu as a user with sudo privileges.
2. Edit the /etc/default/grub file.
3. Set the GRUB_CMDLINE_LINUX value as follows:
4. GRUB_CMDLINE_LINUX="cgroup_enable=memory swapaccount=1"
5. Save and close the file.
6. Update GRUB.

7. $ sudoupdate-grub
8. Reboot your system.

Configure Docker to start on boot

Ubuntu uses systemd as its boot and service manager 15.04 onwards and upstart for versions 14.10 and below.

For 15.04 and up, to configure the docker daemon to start on boot, run

$ sudosystemctl enable docker

For 14.10 and below the above installation method automatically configures upstart to start the docker daemon on boot

Upgrade Docker

To install the latest version of Docker with curl:

$ curl -sSL https://get.docker.com/ | sh

How to run application in docker:

Docker allows you to run applications, worlds you create, inside containers. Running an application inside a container takes a single command: docker run.

Note: Depending on your Docker system configuration, you may be required to preface each docker command on this page with sudo. To avoid this behavior, your system administrator can create a Unix group called docker and add users to it.

Some useful Commands in Docker :-

- The following commands are used to pull an operating system from hub.
- $ sudo docker pull ubuntu
- $sudo docker run -it ubuntu

1. This command is used to populate repository list

- # sudo apt-get update

2. Command to install python 2.7

- # sudo apt-get install python 2.7

3. To install python library

- # sudo apt-get install python-pip

4. Command for website hosting framework

- # sudo pip install flask

5. Command for editor option

- # sudo apt-get install nano

6. Command to check ip

- ifconfig

7. To listing images

- # sudo docker images
- # sudo docker run –it image name

8. To commit changes

- Press ‘ctrl+q’ &‘ctrl+p’ (together)
- # sudo docker ps-l
- #sudo docker commit <container id>

Run a Hello world

$ docker run ubuntu /bin/echo 'Hello world'
Hello world

This will launch the first container!

Advantages of Dockers:-

1. **Simplicity and faster configurations**

An advantage of VMs is allowing the user to run any platform with its own configuration atop the infrastructure of the user. Docker is able to offer the same without the overhead of a VM. Users can take their own configuration, put it into code and deploy it without any fuss.

2. **Increased productivity**

This is where the zero overhead of Docker comes into play. Seeing as a development environment has a very low memory, Docker shows its functionality by not adding to the memory footprint and allowing a few dozen services to run within it.

3. **Rapid Deployment**

Docker manages to reduce deployment to mere seconds. This is due to the fact that it creates a container for every process and does not boot an OS. Data can be created and destroyed without worry that the cost to bring it up again would be higher than affordable.

Input : Commands for installation of docker

Output : Installed docker

Platform : Ubuntu 12.04

Conclusion : Docker is successfully installed and a container is created.

Frequently Asked Questions:-

1. What is the prerequisite for Docker?

2. How does Docker different from VM ware?
3. What is significance of curl command in Docker?