Containers and Docker

Container

- A lightweight OS-level virtualization method along with the stand-alone piece of executable software and NOT a virtual machine. And it can also be called as process with isolation, shared resources, and layered filesystems

VMs VS Containers

- Hypervisors run software on physical servers to emulate a particular hardware system (aka a virtual machine)
- VM runs a fully copy of the operating system (OS)
- Hardware is also virtualized
- Can run multiple applications

EX: Virtual Box, EMU, KVM, etc.

- Run isolated process on a single server or host operating system (OS)
- Can migrate only to servers with compatible OS kernels
- Best for a single application

EX: LXC, Docker, RKT, etc.

Container Rules (Cloud Native Design)

- Containers are immutable
  - Create once, run many instances
- Containers are ephemeral
  - A containers’ life should be only as long as absolutely required
- Containers can be sacrificed
  - Orchestration systems may terminate a container if needed
  - No guarantee of lifespan
  - Don’t store data or logs in containers
- Containers limit resource access
  - Define cgroups for CPU/RAM access
  - Avoid use of ROOT credentials
  - Layered file system helps manage storage
- Cgroups vs namespaces

What is DOCKER, its Platform, Installation, etc.

What is Docker

- Docker is a container management service. The keywords of Docker are **develop, ship** and **run** anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.
- The initial release of Docker was in March 2013 and since then, it has become the buzzword for modern world development, especially in the face of Agile-based projects.

Why Docker

- Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.
- With containers, it becomes easier for teams across different units, such as development, QA and Operations to work seamlessly across applications.
- You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.
- Since Docker containers are pretty lightweight, they are very easily scalable.
- In short: Isolated, Lightweight, Simple, Workflow and Community.

Docker Community

- 1200 Docker Contributors
- 100,000 Dockerized Applications
- 3 to 4 Million Developers using Docker
- 300 Million Downloads
- 32,000 Docker Related Projects
- 70% of enterprises are using Docker

Components of Docker

- **Docker Engine** − It is used for building Docker images and creating Docker containers.
- **Docker Hub** − This is the registry which is used to host various Docker images.
- **Docker Compose** − This is used to define applications using multiple Docker containers.
- **Docker Platform:** Docker Engine + Docker Hub

Docker Engine Consists of (Daemon and CLI)

Docker Daemon

- Builds Images
- Runs and Manages Containers
- RESTful API

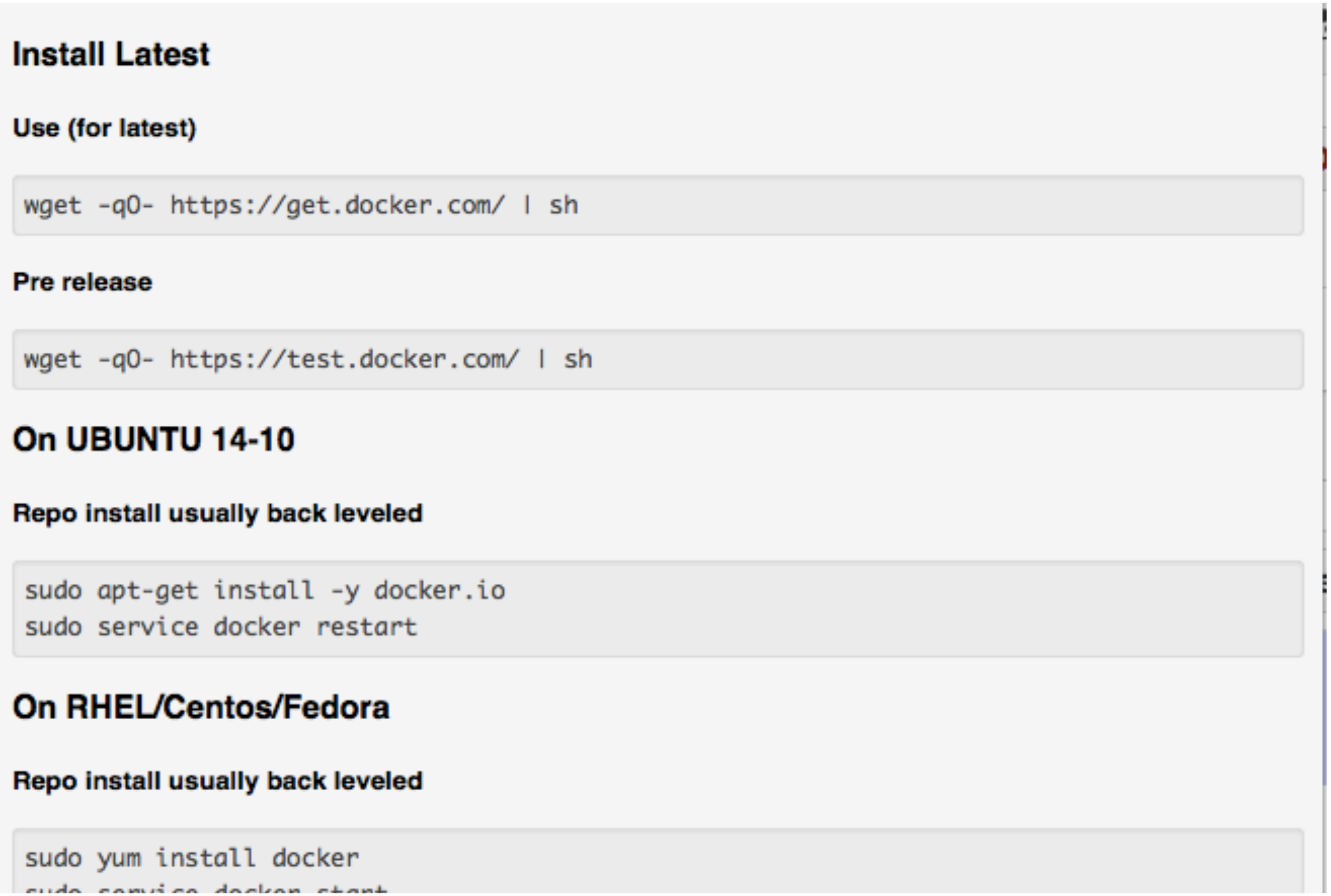
Docker CLI

docker build	# Build an image from a Dockerfile
docker images	# List all images on a Docker host
docker run	# Run an image
docker ps	# List all running and stopped instances
docker stop	# Stop a running instances
docker rm	# Remove an instance
docker rmi	# Remove an image

Docker Hub

- Provides Docker Services
- Library of public images
- Storage for your images
  - Free for public images
  - Cost for private images
- Automated builds(link github/bitbucket repo; trigger build on commit)

Installation



Install Latest

Use (for latest)

```
wget -qO- https://get.docker.com/ | sh
```

Pre release

```
wget -qO- https://test.docker.com/ | sh
```

On UBUNTU 14-10

Repo install usually back leveled

```
sudo apt-get install -y docker.io
sudo service docker restart
```

On RHEL/Centos/Fedora

Repo install usually back leveled

```
sudo yum install docker
sudo service docker start
```

Docker Platform Workflow

- Find an Image on Docker Hub
- Pull an Image from Docker Hub
- Run an Image on Docker Host
- Stop an Instance
- Remove an Instance
- Remove an Image