Chapter 1: Introduction

# 1. The Promise of Docker

-Docker was the 1st widely accessible tool to build on top of a much newer technology called containerization. Docker and Linux containers have a significant impact on a wide range of industry segments that include tools and technologies (Vagrant, KVM, OpenStack…). Docker has altered everyone’s expectation of how a continuous integration and continuous delivery (CI/CD) workflow should function. Most people expects a DevOps pipeline to be fully automated and flow from one step to next without any human intervention, that’s what has given Docker so much.

1.1 Benefits of the Docker Workflow

-**Packaging software in way that leverages the skills developers already have**: Many companies have to create positions for release and build engineers to manage all knowledge and tooling required to create software packages for their supported platforms. Linux tools (rpm, mock, dpkg, pbuilder), Docker wraps up all requirements together into one packing format: Open Container Initiative (OCI) standard

-**Bundling application software and required OS filesystems together in a single standardized image format**

-**Using packaged artifacts to test and deliver the exact same artifact to all systems in all environments**: When dev commit changes to a VCS, a new Docker image can be built, tested and deployed to production without having to be recompiled or repackaged.

-**Abstracting software app from the hardware without sacrificing resources**: Traditional enterprise virtualization solutions (VMware) are used when someone needs to create an abstraction layer between physical layer and software app that run on it, at the cost of resources. A container is just another process that talks directly to underlying Linux kernel and can utilize more resources, up until system or quota-based limits are reached.

# 2. What Docker isn’t

-Enterprise virtualization platform (VMware, KVM)

+VM contain a complete OS, running on top of a hypervisor that is managed by underlying host OS. Hypervisors create virtual hardware layers that make it possible to run additional OS on top of a single physical computer system -> can run many VMs with different OS on a single host.

+Containers: host and containers share the same kernel. -> containers utilize fewer system resource but must be based on the same underlying OS.

-Cloud platform (OpenStack, CloudStack)

+Docker only handles deploying, running, and managing containers on preexisting Docker hosts. It doesn’t allow to create new host systems (instances), object store, block storage, many other resources that are often managed with cloud platform

-Configuration management (Puppet, Chef)

-Deployement framework (Capistrano, Fabric)

+Docker can’t be used to automate a complex deployment process by itself.

+Docker and other Linux container toolsets (Kubernetes) provide an interface for deployment, the method required to deploy containers will be consistent on all hosts.

-Development environment (Vagrant)

-Workload management tool (Mesos, Kubernetes, Swarm)

-Linux containers provide a way to run software in a controlled and isolated environment, while the easy-to-use CLI tooling and container image standard that Docker introduced made working with containers easier and ensured that there was a repeatable way to build software across the whole fleet.

3. Important Terminology

-**Docker client**: docker command used to control most of Docker workflow and talk to remote Docker servers.

-**Docker server**: dockerd command used to start Docker server process that builds and launches containers via a client.

-**Docker or OCI images**:

+Consist of one or more filesystem layers and some important metadata that represent all the files required to run containerized app. A single image can be copied to numerous hosts. An image has a repository address, a name, a tag. Tag is used to identify a particular release of an image.

+Docker image is any image that is compatible with Docker toolset, OCI image is specifically an image that meets the Open Container Initiative standard and is guaranteed to work with any OCI-compliant tool.

-**Linux container**: a container that has been instantiated from a Docker or OCI image. A specific container can exist only once, but you can create multiple containers from the same image. Docker container is a misnomer since Docker simply leverages the OS’s container functionality.