D.1. Why Docker?

* Docker's commercial solutions provide an out of the box **CaaS environment** that gives **IT Ops** teams **security** and **control** over their environment, while enabling **developers** to **build app**lications in a **self service** way
* With a clear separation of concerns and robust tooling, **organizations** are able to innovate **faster, reduce costs and ensure security**

**i.e**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Developer** | **IT Ops** | **General** |
| **Agility** | have freedom to define environments, and the ability to create apps | can deploy apps faster, allowing the business to outpace competition | - |
| **Control** | own all the code from infrastructure to app | have the manageability to standardize, secure, and scale the operating environment, while reducing overall costs to the organization | - |
| **Portability** | - | - | Docker gives teams the choice to leverage any infrastructure whether in the cloud, on VMs or baremetal servers allowing companies to make the best business decision for them. |

D.2. Docker Use-cases

1. **CI/CD**

* CI/CD merges **development with testing**, allowing developers to **build** code **collaboratively**, **submit** it the **master** branch, and **checked for issues**.
* Since **Docker** can **integrate** with tools like **Jenkins** and **GitHub**, developers can **submit code in GitHub**, **test** the code and **automatically trigger a build** using **Jenkins**, and once the **image** is **complete**, images can be **added to Docker registries**.
* This **streamlines the process**, **saves times** on build and set up processes, all while allowing **developers** to **run tests in parallel** and **automate them** so that they can **continue to work on other projects** while **tests are being run**.

1. **DevOps**

* Today 44% of enterprises are looking to adopt a DevOps initiative within their organization. This cultural shift is geared towards **tearing down** the **traditional barrier** that has existed between **Developer** teams and **IT operations** teams. The goal is to help enable **DevOps** within the enterprise **via** the **Docker platform**.
* **The Docker CaaS platform** enables **developers** to **build** applications in a **self service** manner and **select** from **image content that the IT operations team** has **deemed okay** for developer use. Developers can then use these **images** to **create new applications**, **quickly** and **securely**.
* **Docker’s enterprise tool, Docker Datacenter**, delivers a **Containers as a Service** (CaaS) environment that **deploys on-premises** and is chock full of enterprise-grade security feature like **role-based access controls**, **image signing** and **image scanning** giving IT operations teams ability to **secure** and **manage** their **env**ironment.

1. **Infrastructure Optimization**

* **VMs i**nclude a **guest operating** within each machine, making them **heavyweight** and **eating up valuable storage capacity.**

**Docker containers** are **lightweight runtimes**, and **include** only **what’s necessary** to **run** your **app**lications.

**Each container** running on a **Docker engine**, which installs **on a host**, shares the **same Linux kernel**, with **no guest operating system** **within each container**

* **Portability** is another factor. Docker provides solutions that allow enterprises to **leverage Docker containers** across **any infrastructure** or **application environment type**

D.3. Docker for Linux Installation and verification

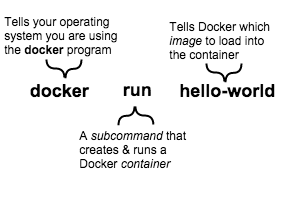
* Installation

<https://docs.docker.com/engine/installation/linux/centos/>

<https://docs.docker.com/engine/getstarted/linux_install_help/>

* Verification of installation
* **docker version**
* **docker ps**
* **docker run hello-world**

D.4. Images and containers



* **Image** is a **filesystem** and **parameters** used **at runtime**
* It **doesn’t have a state** and **never changes**
* A **container** is a **running instance** of an **image**
* An **image** can **start software** as complex as a database, **wait for you** (or someone else) **to add data**, **store the data for later use**, and then **wait for the next person**
* When you ran the command “docker run hello-world”, the Docker Engine:

1. checked to see if you had the hello-world software image
2. downloaded the image from the Docker Hub (more about the hub later)
3. loaded the image into the container and “ran” it

* Who built the ***“hello-world”*** software image though?

Docker Engine lets people (or companies) **create and share software through Docker images.**

* Using Docker Engine, you don’t have to worry about whether your computer can run the software in a Docker image — **a Docker container can always run it.**

D.5.