Lecture Notes: *Docker*

- What is Docker?
  - What are containers?
  - Why use containers?
  - How do containers fit into the DevOps lifecycle?

What is Docker?

"Docker an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production."

What is Containerisation?

Containerisation is a form of operating system virtualisation

- Applications are run in isolated user spaces called *containers*

What are containers?

A **process** on your machine that has been isolated from all other processes on the host machine

Container Image: has the isolated filesystem, and everything needed to run an application

- Application dependencies, configurations, scripts, etc.
- Container configurations, environment variables, etc.

Why use containers?

Immutable Infrastructure

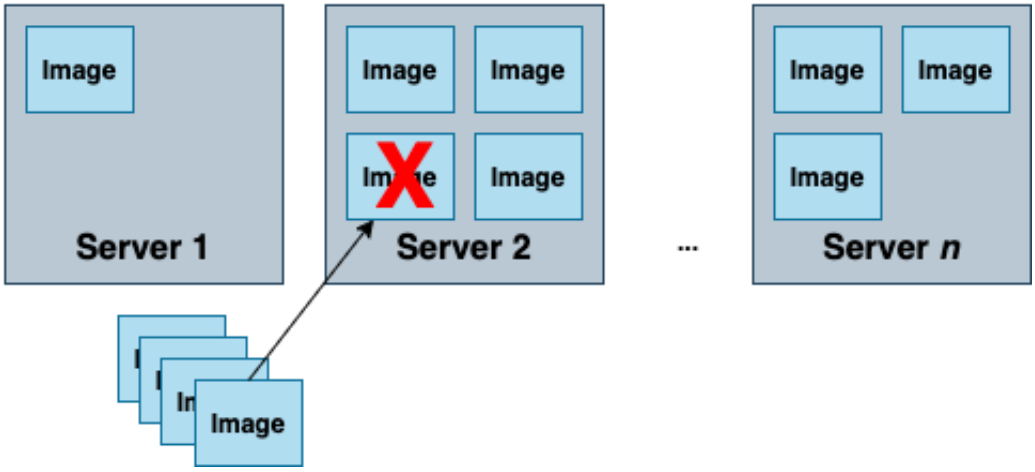
No rigorous or standardized definition of immutable infrastructure exists yet, but the basic idea is that you create and operate your infrastructure using the programming concept of immutability.[1]

When we can quickly rebuild and re-create our applications and environments on demand, we can also quickly rebuild them instead of repairing them when things go wrong. Although this is something that almost all large-scale web operations do (i.e., more than one thousand servers), we should also adopt this practice

even if we have only one server in production.[2]

"Treat servers like cattle, not pets."

Containers are ephemeral: short lived, if they die or go away we just replace them:



Containers vs. Virtual Machines

Containers and VMs used together provide flexibility in deploying and managing applications

What can you do with Docker?

Build Images	Share Images	Run Images
Consistently packag everything the app needs to run	Easily share the images to runtimes in the cloud or on your local machine	Easily and consistently execute your applications
CI/CD	**Different Versions**	**Roll Forward**
Consistently test and deploy your code to different environments e.g. Stage, UAT, Production	Easily run different versions of your software without installation	When a defect is found, no need to patch or update, can just build a new image

Are Containers Secure?

Security

Is Docker Secure?

Containers are secure by default. Everything is opt-out.

- Can sign container images
  - Docker Content Trust
- Can scan container images
  - Docker Trusted Registry

Orchestration

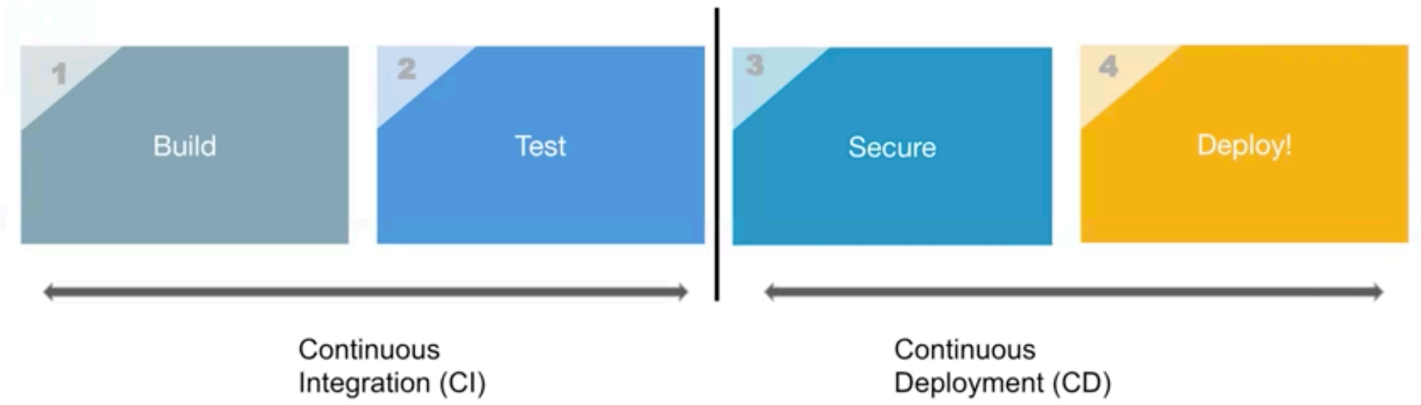
Kubernetes and Docker Swarm are the most common Orchestrators for automating the managing maintenance of containers.

DevOps and Containers

DevSecOps

CI/CD and Docker Security

1. Image can be uploaded to DTR and scanned
2. If no vulnerabilities, image can be moved on to next stage in pipeline
3. Then final sign off, from Security



Summary of Containers

Applications of Docker [3]

1. Develop with Docker
   - How to create a docker image
   - How to create a docker container
2. Docker and CI/CD
   - Secure your container in your pipeline

3. Deploy and manage your containers with Swarm/Kubernetes
    - The process of integrating containerisation with you CI/CD pipeline

Docker Core Pillars

Choice

- Any application, new apps or legacy apps
- Any infrastructure, on premesis, on the cloud, on the edge
- Industry standard runtime and image

Security

- Built into every layer of the container stack
- Integrated into software supply chain
- Image chain of custody
- Fine-grained access control and governance

Agility

- Reduce time taken to release and patch
- Integrate into existing toolchains and pipelines
- Reduce overhead on infrastructure and operations
- Accelerate the migration to contaiers

---

1. Immutable Infrastructure ↩
2. Devops Handbook ↩
3. Docker Tutorial ↩