DevSecOps Essentials

Section 2:Secure Software Onboarding

Lesson 2.3 : Secure Containerization

Containers

- **Containers**: Specially formatted files that contain executable application programs, modules, and the code libraries that they depend upon.
- They isolate an application from other applications and from the operating system and hardware.
- Containers help ensure that the versions of libraries within the container do not mandate patching and upgrades that might affect other applications in other containers.
- In virtual environments, the libraries and dependencies of an application were oftentimes shared with all of the other applications on a particular virtual server.

Use Case: Docker

- While there are many container engines to choose from, our use case is Docker.
- Process Restrictions use root/non-root dichotomy. This makes it difficult to provoke system level damages during an intrusion, even if the intruder manages to escalate to root within a container because the container capabilities are fundamentally restricted.
- Device and File Restrictions further reduces the attack surface by restricting access by containerized applications to the physical devices on a host, through the use of the device resource control groups (cgroups) mechanism.
- Container images are ephemeral. Each container has its own file system and can not write to the host file system unless writes are committed. Committing changes tracks and audits revisions made to the base images as a new layer which can then be pushed as a new image for storage in Docker Hub and run in a container.
- The audit trail is important in providing information to maintain compliance. It also allows for a fast and easy rollback to previous versions if a container has been compromised or a vulnerability introduced.

Docker Hub: The World's Largest Community of Container Images

Docker Hub is the world's largest public repository of container images with an array of content sources including container community developers, open source projects, and independent software vendors (ISV) building and distributing their code in containers.

Docker Hub allows you to:

- Search and browse for millions of container images
- View image popularity, vendor source and certification to inform your selection
- Access free public repositories to store your images and share with the community
- Choose a subscription plan for private repositories to limit access to your images
- Use Autobuilds and Webhooks for easy integration into your DevOps pipeline

Docker Hub

PUBLIC

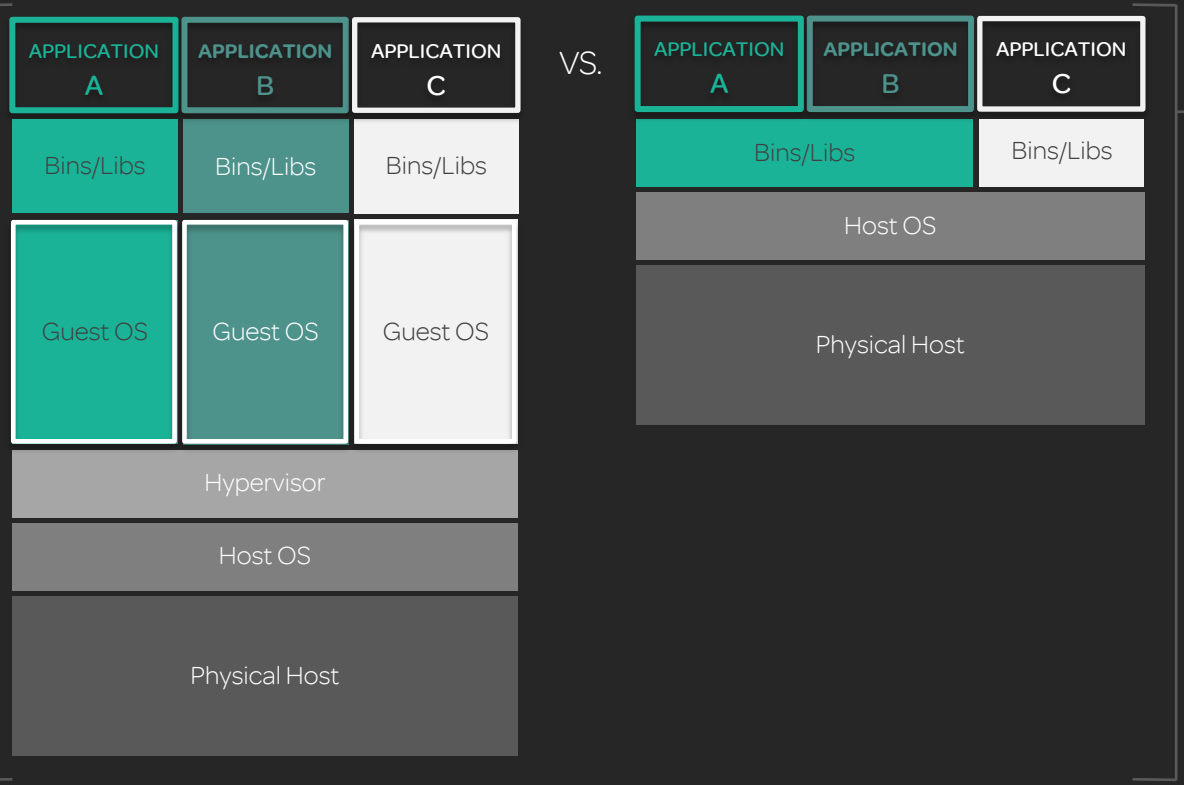
PRIVATE

OFFICIAL

- While Docker Hub is a public repository, it contains public, private, and official images.
- Public images are shared amongst a broad community.
- Private images might be limited to a single organization or even project within an organization.
- Official repositories are certified repositories from independent software vendors (ISV's) and Docker contributors such as Canonical, Oracle, RedHat, and others.

VIRTUAL MACHINES

Virtual machines utilize a host OS with their own libraries. A hypervisor is used to facilitate the installation and running of guest virtual servers. The guest servers have their own image of an operating system and libraries. Applications within each virtual server rely on the version of the libraries and the guest operating system instance.

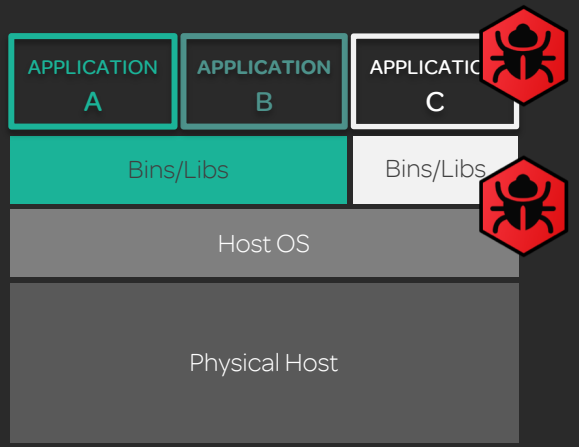


CONTAINERS

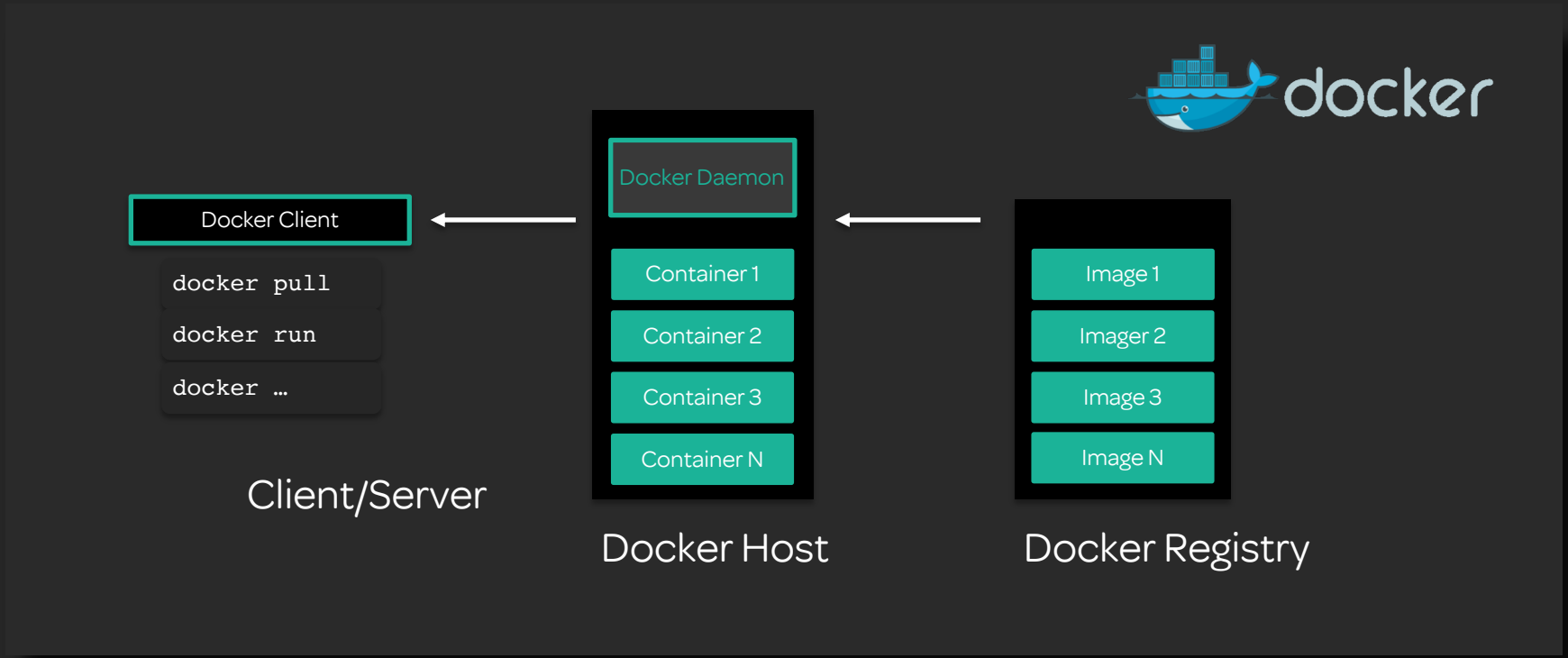
Containers rely on isolation techniques like control groups and namespaces to separate applications. They do not have a hypervisor and can share libraries if needed. Containers share the physical host's kernel, so they do not require an operating system. Container systems have processes in place to prevent a container from making potentially dangerous changes to the kernel.

Docker and Malware

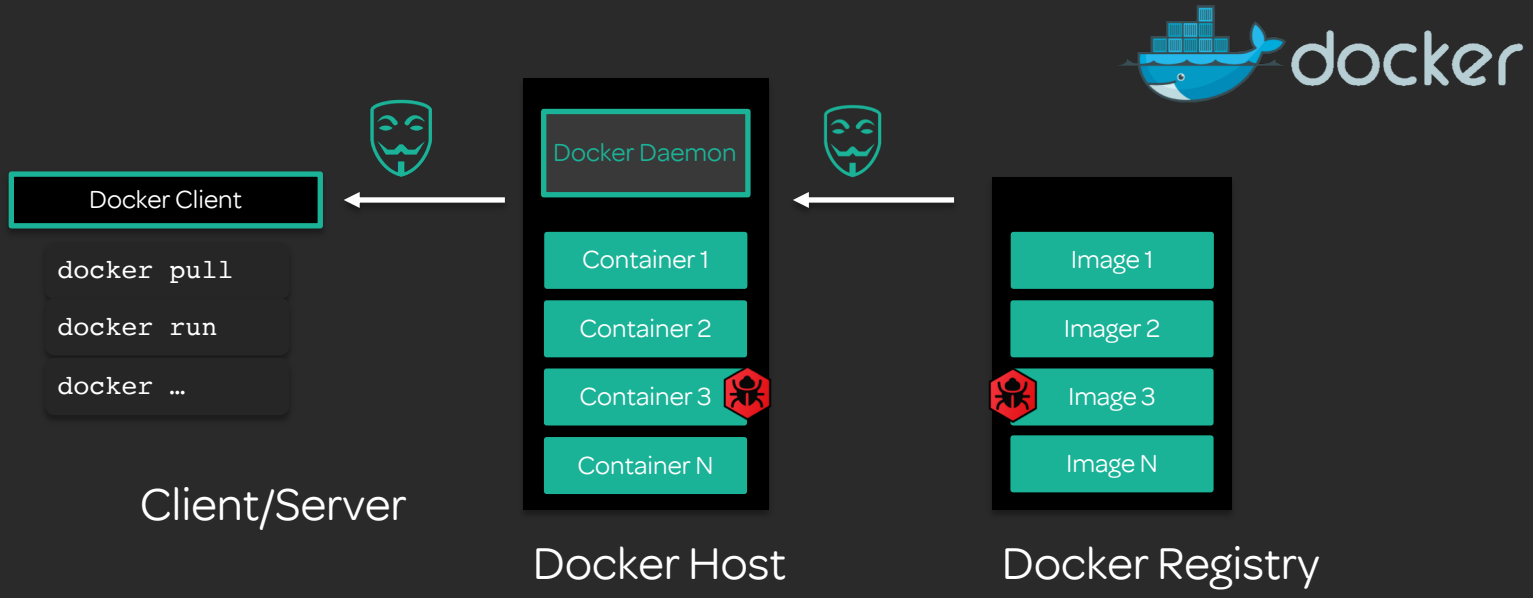
Just as a repository can contain malicious source code or infected libraries, Docker containers may likewise be tainted. When they are replicated broadly throughout on-premise and off-premise infrastructures, they can carry the malware with them and compromise the integrity of production systems.

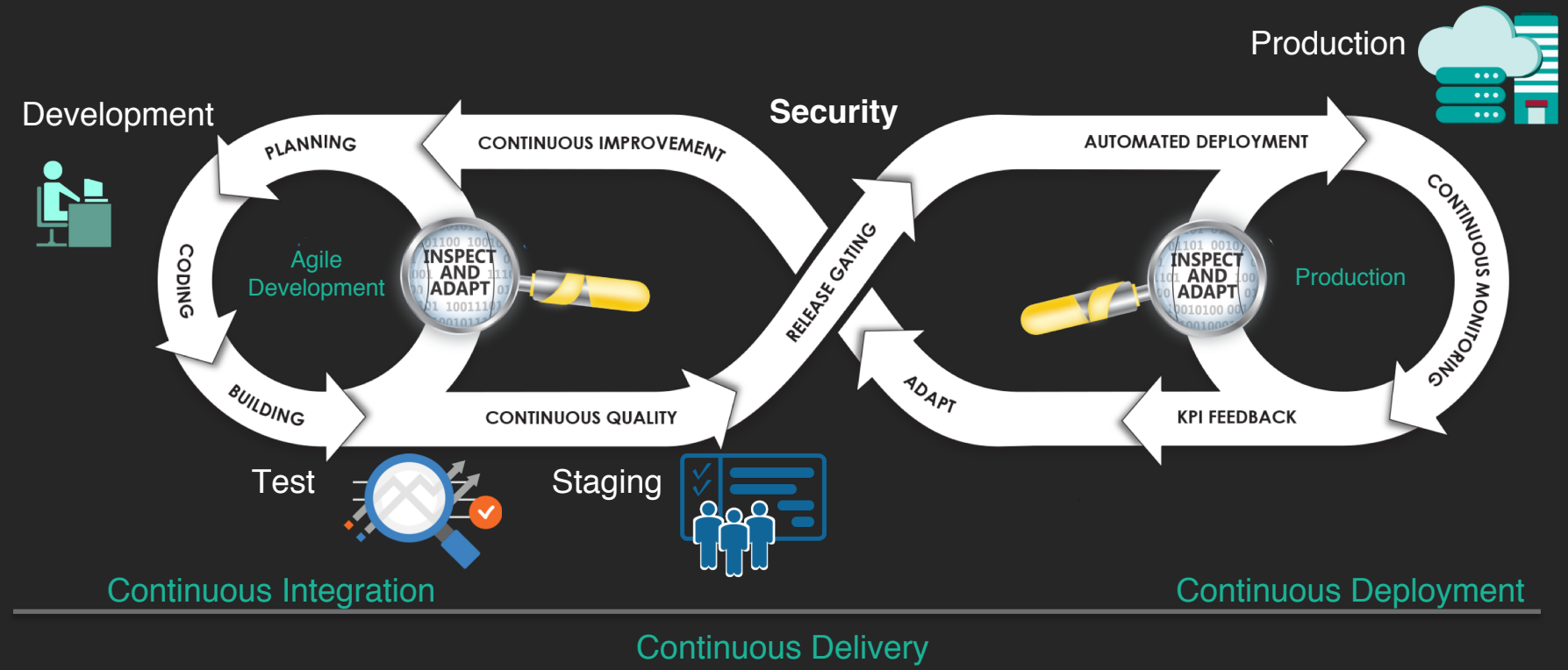


Docker Architecture



Docker Architecture (cont.)




Development
Production
Security
PLANNING
CONTINUOUS IMPROVEMENT
AUTOMATED DEPLOYMENT
CODING
Agile Development
INSPECT AND ADAPT
RELEASE GATING
INSPECT AND ADAPT
Production
CONTINUOUS MONITORING
BUILDING
CONTINUOUS QUALITY
ADAPT
KPI FEEDBACK
Test
Staging
Continuous Integration
Continuous Deployment
Continuous Delivery