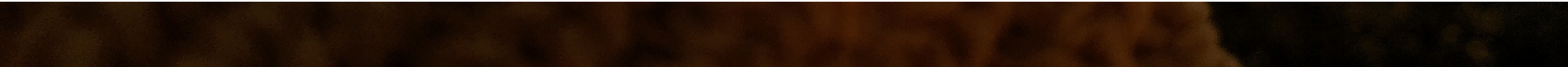
INTRODUCTION TO DOCKER

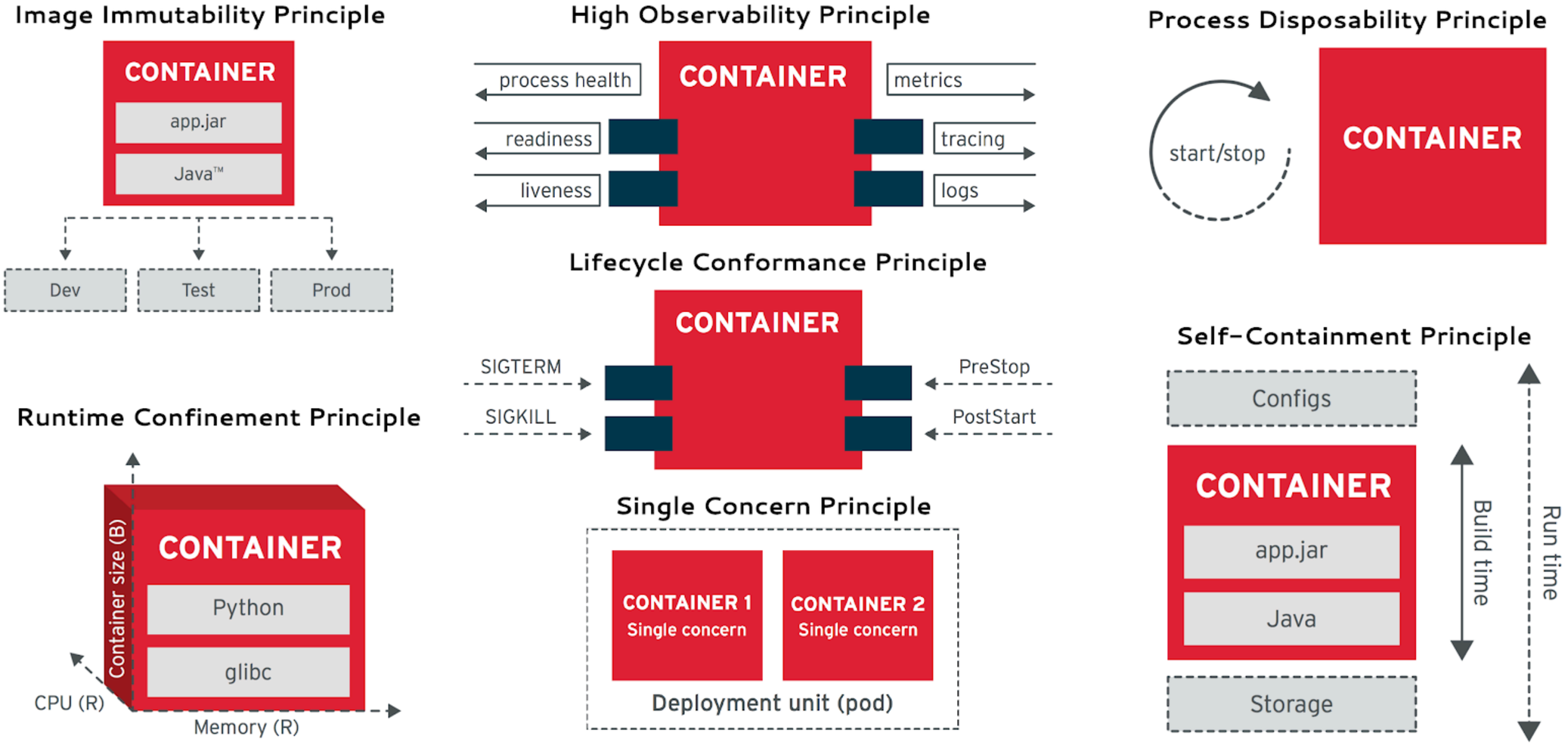
INTRODUCTION TO DOCKER

Container Principle

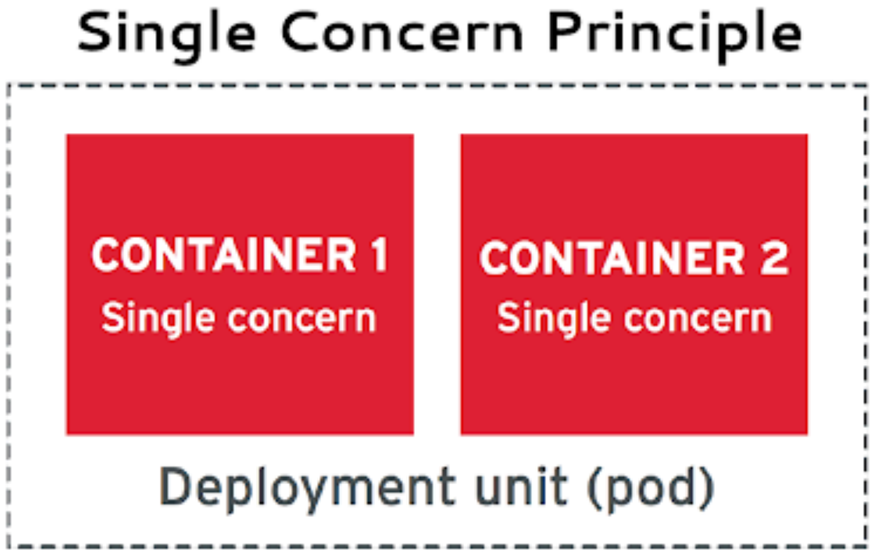


CONTAINER PRINCIPLES

CONTAINER PRINCIPLES



SINGLE CONCERN PRINCIPLE (SCP)



In many ways, this principle is similar to the single responsibility principle (SRP) from SOLID, which advises that a class should have only one responsibility.

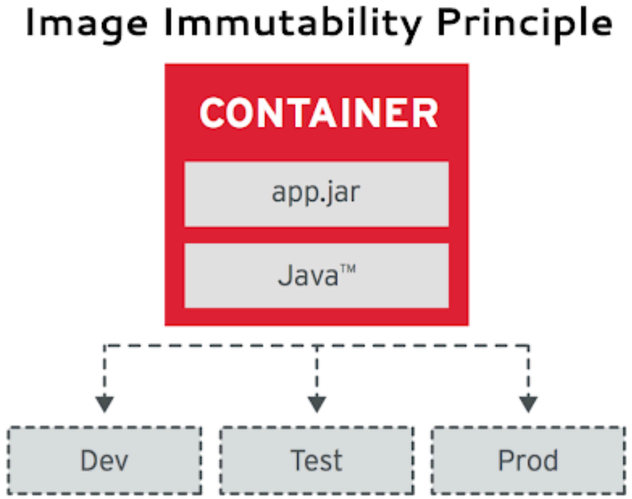
HIGH OBSERVABILITY PRINCIPLE (HOP)

Containers provide a unified way for packaging and running applications by treating them like a black box. But any container aiming to become a cloud-native citizen must provide application programming interfaces (APIs) for the runtime environment to observe the container health and act accordingly.

LIFE-CYCLE CONFORMANCE PRINCIPLE (LCP)

The HOP dictates that your container provide APIs for the platform to read from. The LCP dictates that your application have a way to read the events coming from the platform. Moreover, apart from getting events, the container should conform and react to those events.

IMAGE IMMUTABILITY PRINCIPLE (IIP)

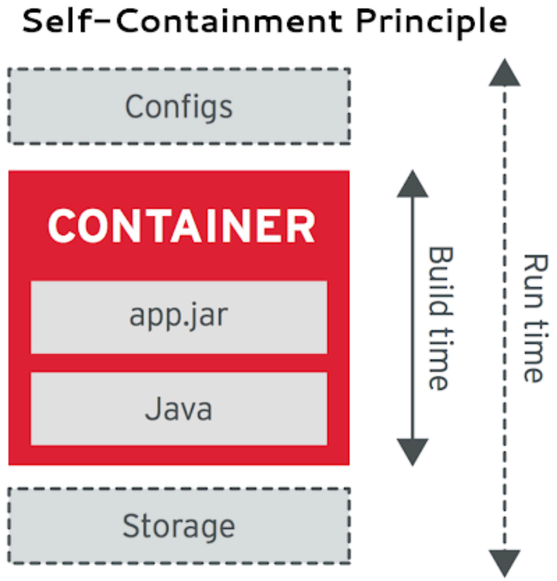


Containerized applications are meant to be immutable, and once built are not expected to change between different environments. This implies the use of an external means of storing the runtime data and relying on externalized configurations that vary across environments, rather than creating or modifying containers per environment.

PROCESS DISPOSABILITY PRINCIPLE (PDP)

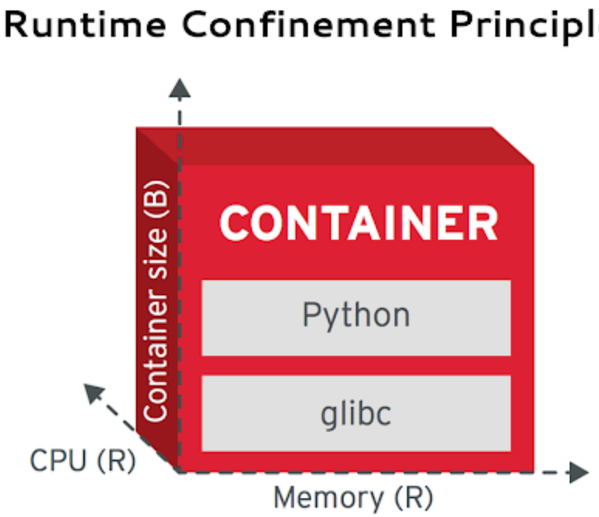
One of the primary motivations for moving to containerized applications is that containers need to be as ephemeral as possible and ready to be replaced by another container instance at any point in time.

SELF-CONTAINMENT PRINCIPLE (S-CP)



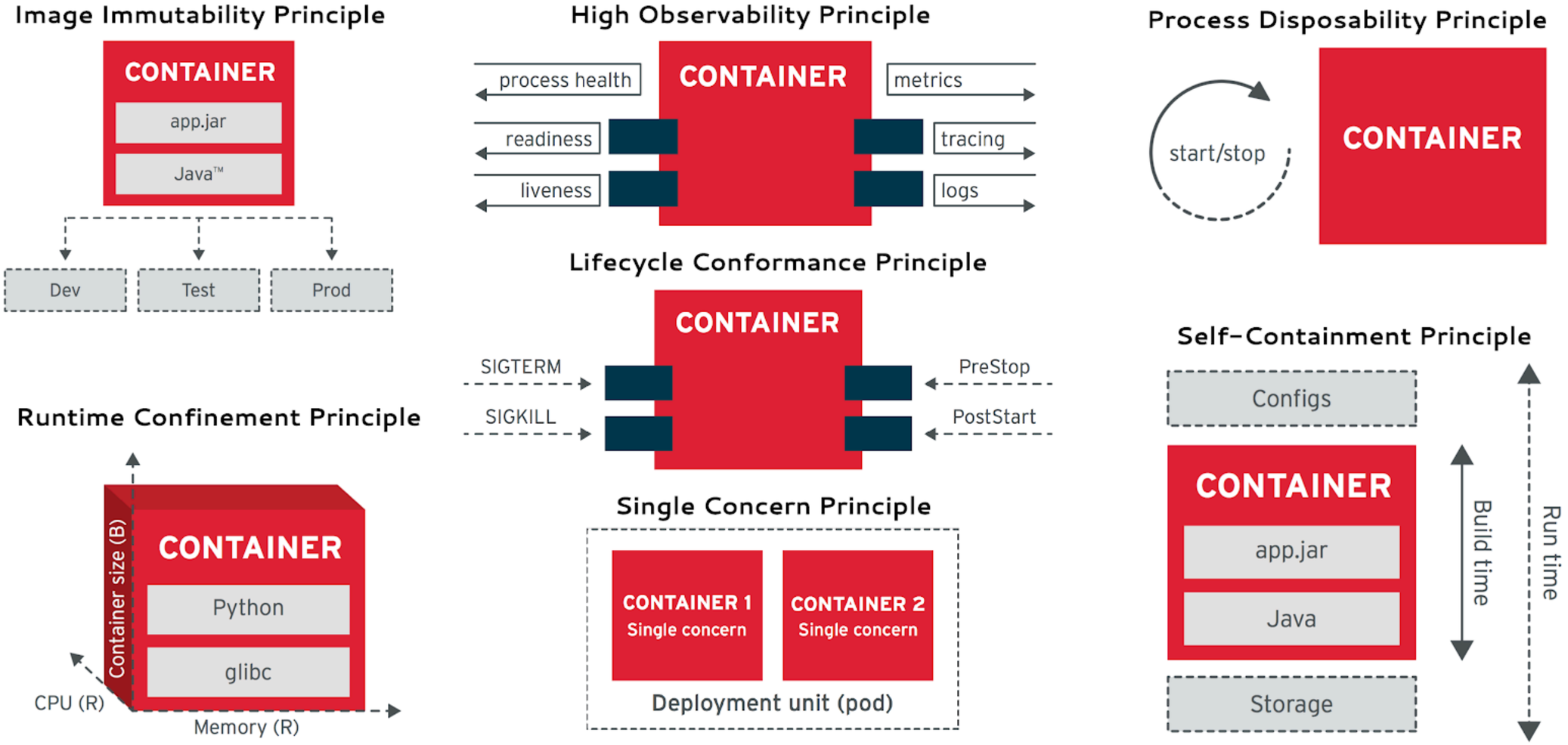
This principle dictates that a container should contain everything it needs at build time. The container should rely only on the presence of the Linux® kernel and have any additional libraries added into it at the time the container is built. In addition to the libraries, it should also contain things such as the language runtime, the application platform if required, and other dependencies needed to run the containerized application.

RUNTIME CONFINEMENT PRINCIPLE (RCP)



This RCP principle suggests that every container declare its resource requirements and pass that information to the platform. It should share the resource profile of a container in terms of CPU, memory, networking, disk influence on how the platform performs scheduling, auto-scaling, capacity management, and the general service-level agreements (SLAs) of the container.

CONTAINER PRINCIPLES



THANK YOU