

# Introduction to containers and Docker

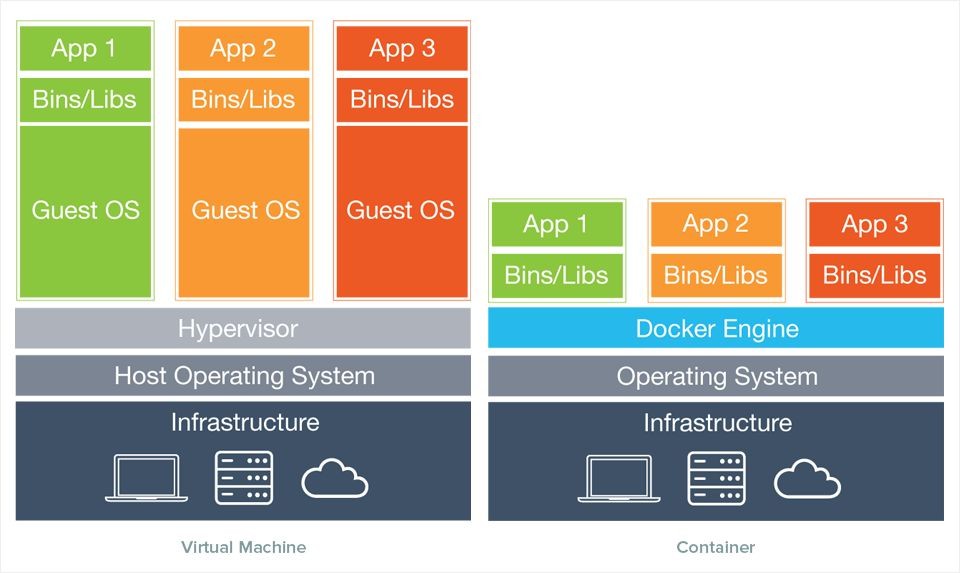
February 24, 2025

## Table of Contents

1. Introduction to containers and Docker (20 min)
2. Docker tutorial / live demo (50 min)
3. Overview of containers in HPC (15 min)

## Containers

* Isolated environments to run applications/services
* Images include all software dependencies
* Prescriptive, portable, easy to build, quick to deploy



Introduction to Containers and Docker 3

## Key terms

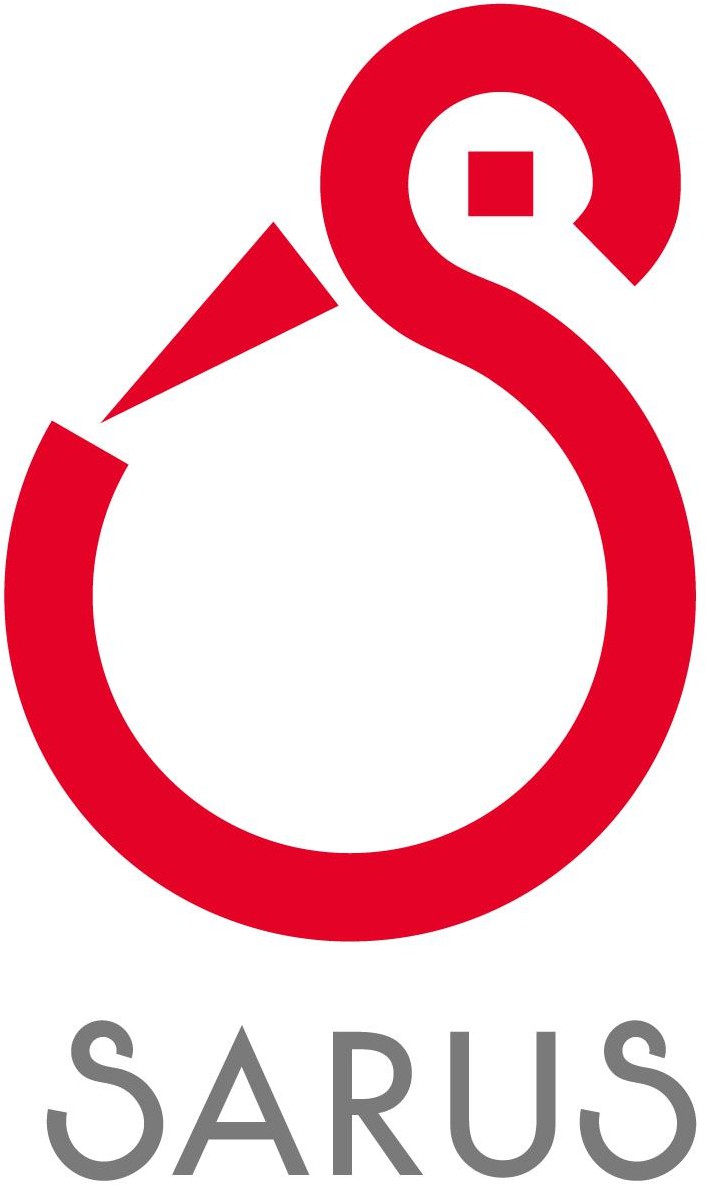
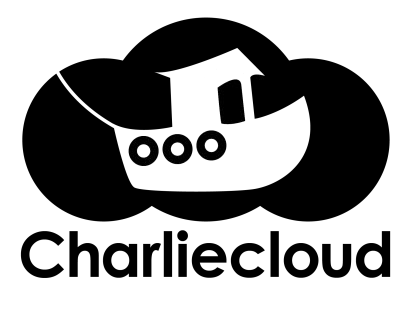
* **Image**: standalone, executable package that includes everything needed to run a piece of software (code, runtime libraries, configuration files). Provides the filesystem and metadata (e.g. environment variables, initial working directory) for a container.
* **Container**: a process isolated from the rest of the system through abstractions created by the OS. The level of isolation can be controlled, allowing access to host resources. Its filesystem content comes from an image.
  + Can be thought as the runtime *instance* of an image: what the image becomes in memory

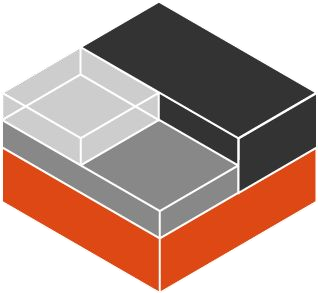
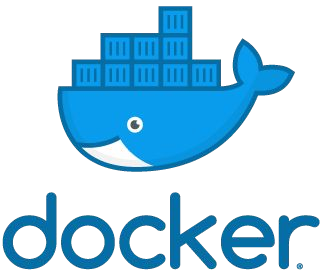
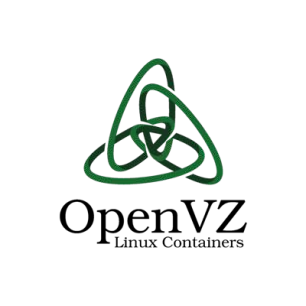
when actually executed.

## Linux containers ecosystem

* Linux containers rely on abstraction features (*namespaces*1) provided by the kernel
* Different design decisions and use cases gave rise to several solutions:

**HPC focused**

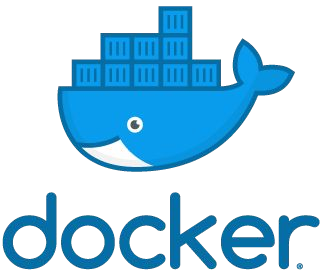


**LXC**

**Singularity**

****

## Docker

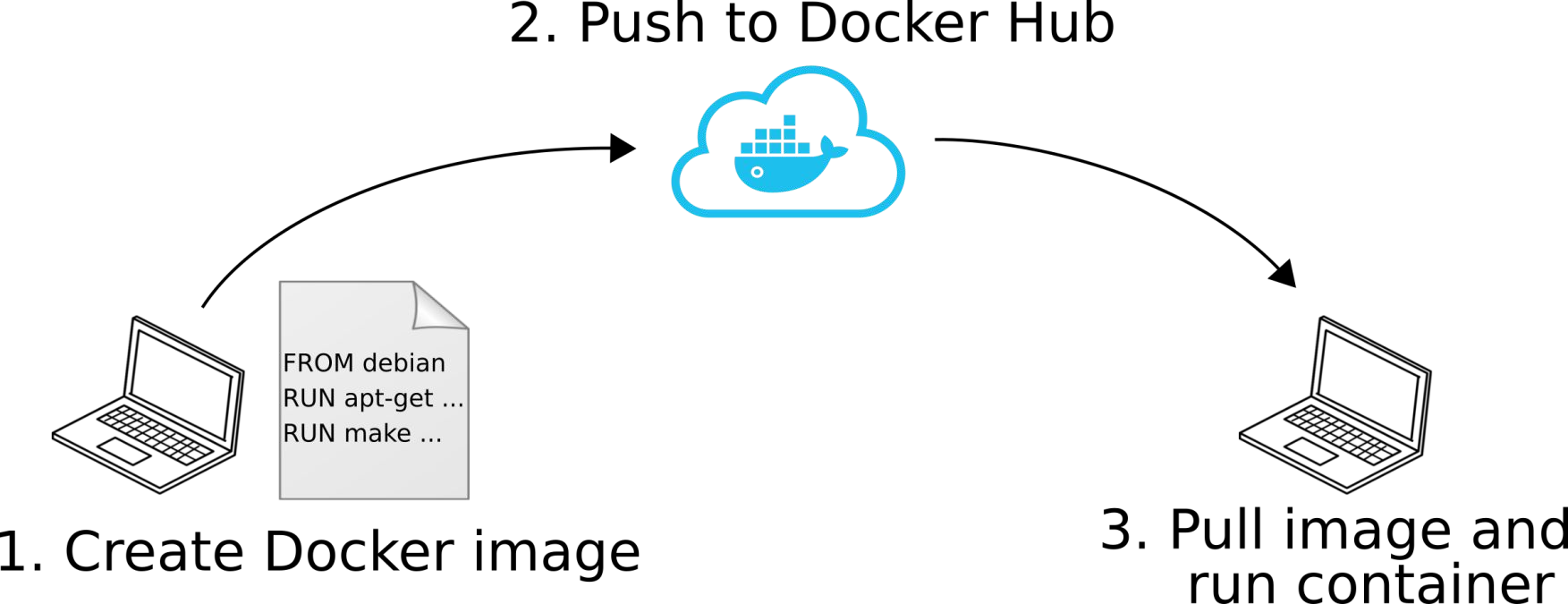
* Extremely popular container implementation
* Easy to use authoring tools
  + Container images are created from recipe-like files
  + Images can be named, tagged and built on top of other images
* Cloud-based image distribution strategy
  + Several remote registries available (e.g. Docker Hub)
  + Client includes facilities to authenticate, push and pull images

## Docker workflow

1. An image is created locally from a Dockerfile
2. Push (i.e. upload) the image to a remote registry

DockerHub is the public registry maintained by the Docker company

1. Pull (i.e. download) the image on a target machine and run the container



## So… how are containers useful?

* + Containers give the possibility to create (scientific) applications that are:

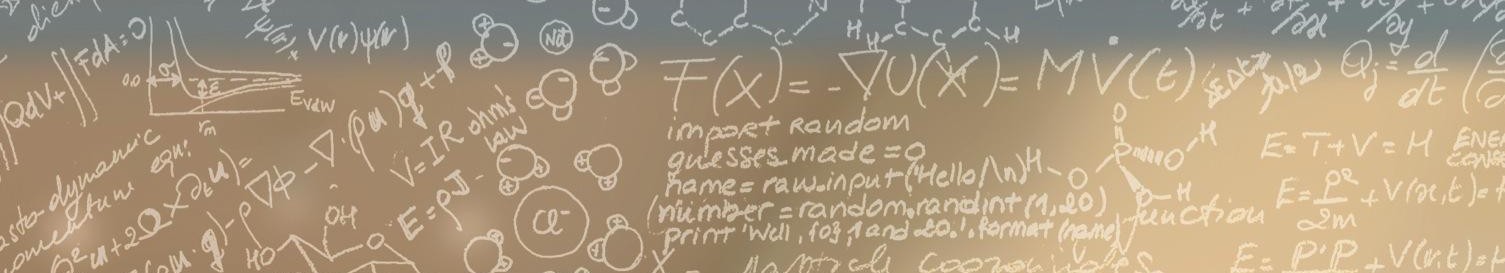
1. Portable
2. Prescriptive
3. Easy to deploy
4. Easy to test

# Commands

**Docker cheatsheet**

**docker pull <user/image:tag> docker run <image:tag> <command> docker run –it <image:tag> bash**

|  |  |  |
| --- | --- | --- |
| **docker** | **run <image:tag> mpirun –n** | **2** |
| **docker** | **images** |  |
| **docker** | **build –t <user/image:tag>** | **.** |
| **docker** | **login** |  |
| **docker** | **push <user/image:tag>** |  |



# Thank you for your attention.