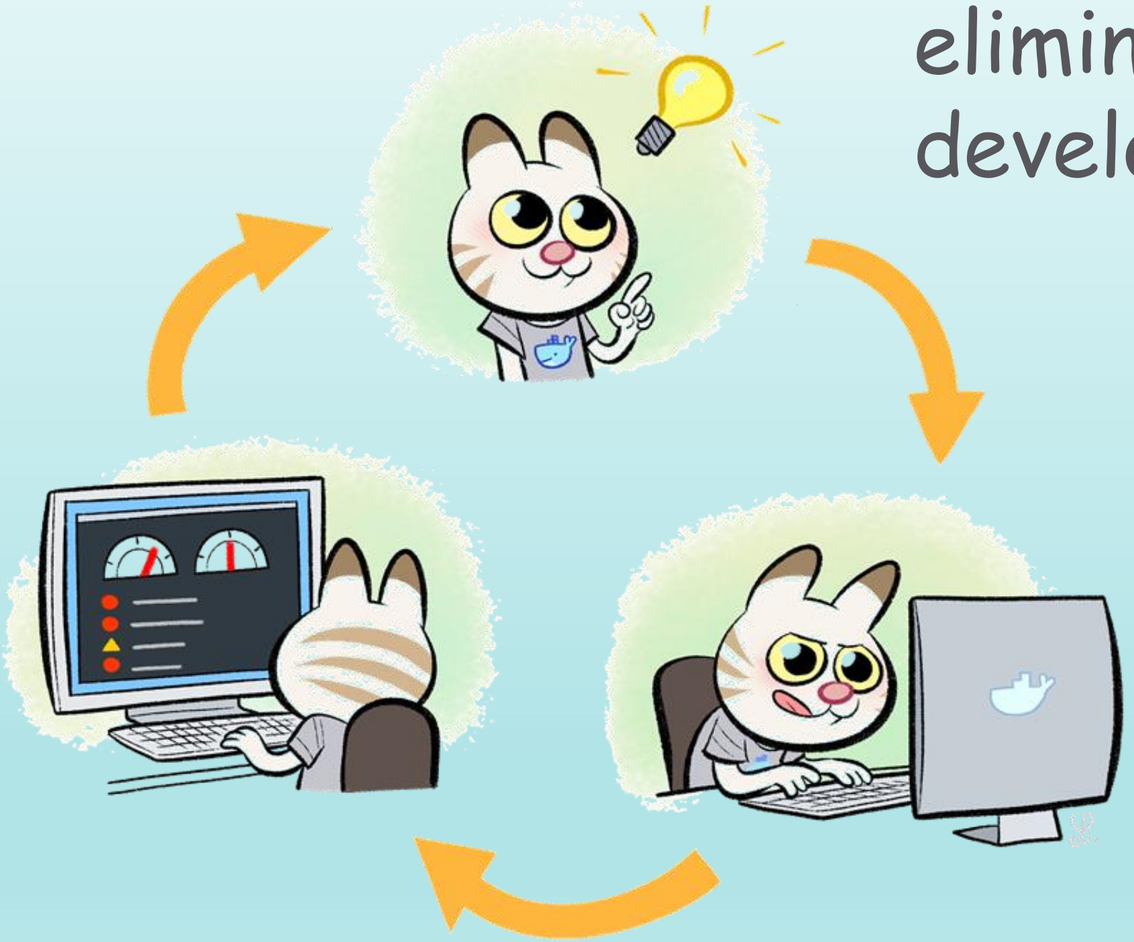
Why Containers?

Why Docker?

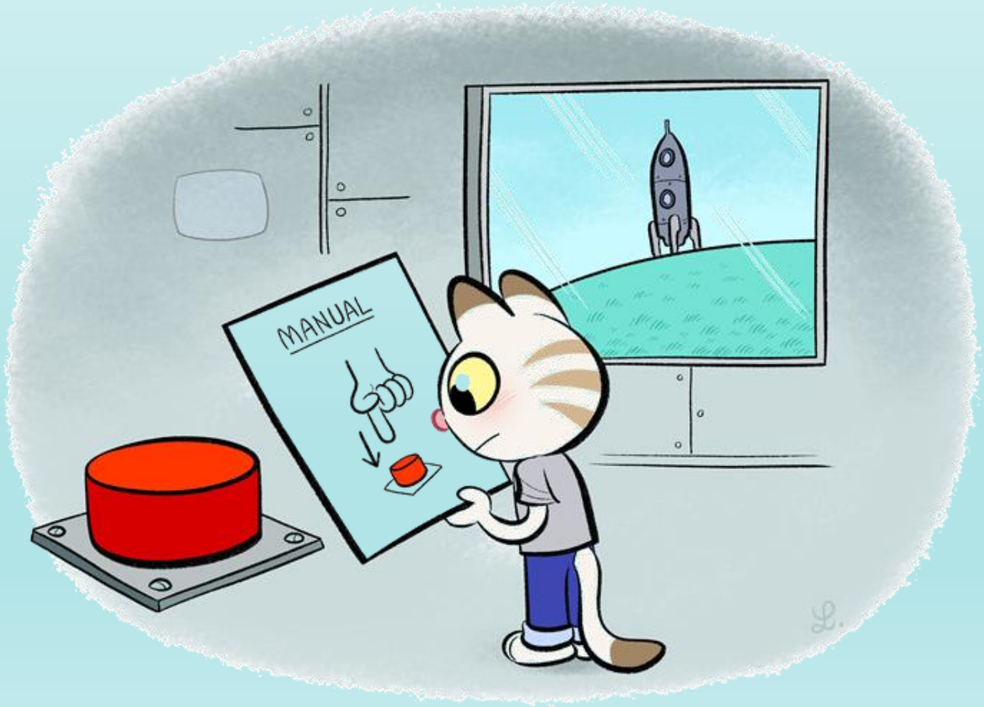


eliminate friction in the development cycle

Why Docker?

Easily adapts to your working environment

Make the powerful simple



Why Docker?

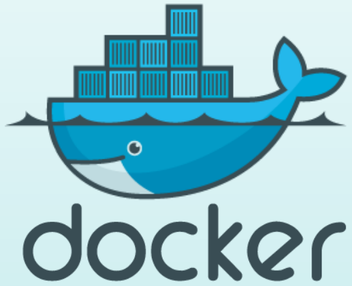
Docker Containers Are Popular

Develop an app using Docker containers with any language and any toolchain.

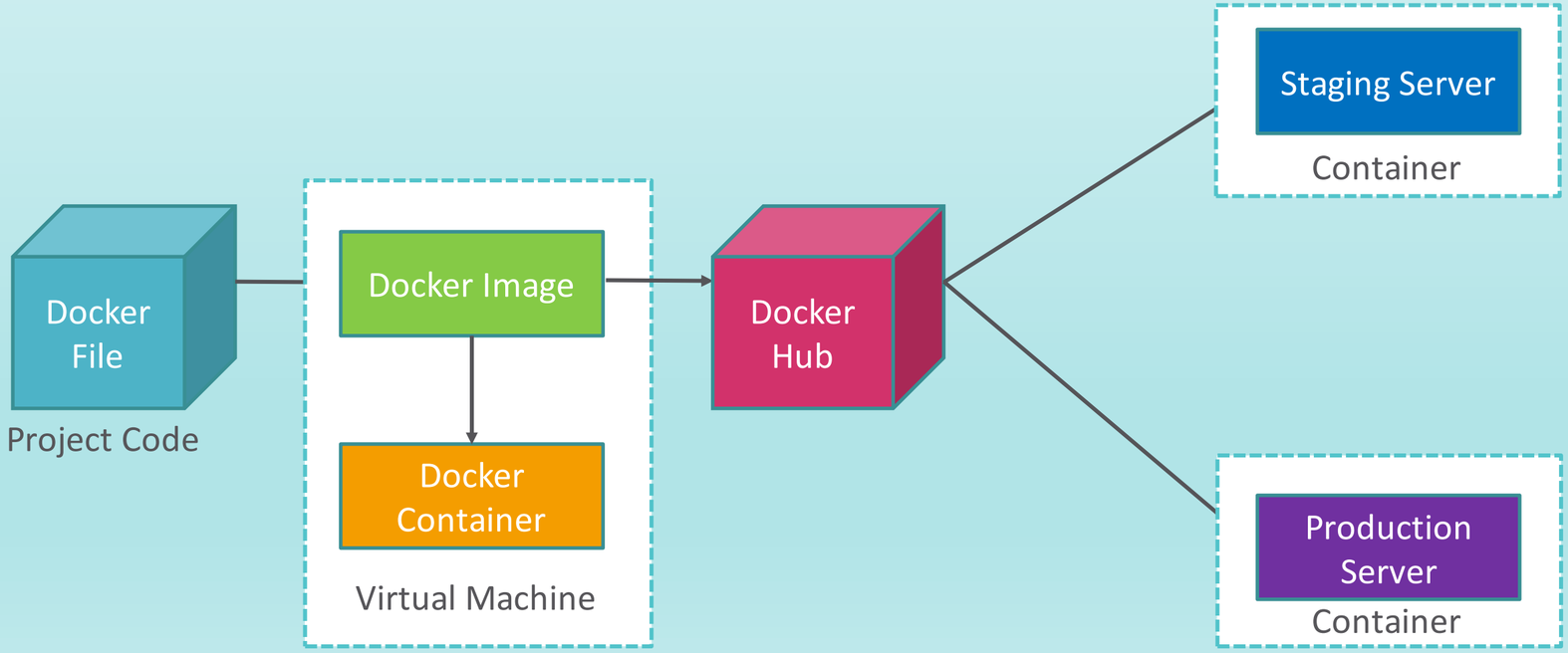
Ship the "Dockerized" app and dependencies anywhere - to QA, teammates, or the cloud - without breaking anything.

Scale to 1000s of nodes, move between data centers and clouds, update with zero downtime and more.

Docker In A Nutshell



- Docker file builds a Docker image which contains all the project's code
- You can run that image to create as many docker containers as you want
- The created Images can be uploaded on Docker hub from where the image can be pulled and built in a container



What is Container?

- Container have an OS
- In other words, it is packaged with everything required to make the software run, except the OS, hence making the application portable
- All you need is libraries and settings to make the software work on any system
- It makes the container efficient, self-contained, lightweight system and guarantees that the packaged software will always run, regardless of where it is deployed

Why Containers Matter ?

	Physical Containers	Docker Container
Content Agnostic	• The same container can hold almost any type of cargo	• Can encapsulate any payload and its dependencies
Hardware Agnostic	• Standard shape and interface allow same container to move from ship to train to semi-truck to warehouse to crane without being modified or opened	• Using operating system primitives (e.g. LXC) can run consistently on virtually any hardware—VMs, bare metal, openstack, public IAAS, etc.—without modification
Content Isolation and Interaction	• No worry about anvils crushing bananas. Containers can be stacked and shipped together	• Resource, network, and content isolation. Avoids dependency
Automation	• Standard interfaces make it easy to automate loading, unloading, moving, etc.	• Standard operations to run, start, stop, commit, search, etc. Perfect for devops: CI, CD, autoscaling, hybrid clouds
Highly efficient	• No opening or modification, quick to move between waypoints	• Lightweight, virtually no start-up penalty, quick to move and manipulate
Separation of duties	• Shipper worries about inside of box, carrier worries about outside of box	• Developer worries about code. Ops worries about infrastructure.