Docker for People

A brief and fairly painless introduction to Docker

Friday, November 17th 11:00 - 11:45

Greg Gómez
Sung-Hee Lee
The University of New Mexico
IT

Docker for People

Agenda:

Greg: Theory

Sung-Hee: Practice (Demo)

Note

We’re (mainly) php developers, so we’ll be using LAMP (Linux, Apache, MySQL, php) for ~~many~~ **all** of our examples.

Not all version numbers in this presentation are real; some are entirely fictional.

What Is Docker?

Docker is the world's leading software container platform. Developers use Docker to eliminate "works on my machine" problems when collaborating on code with co-workers. Operators use Docker to run and manage apps side-by-side in isolated containers to get better compute density. Enterprises use Docker to build agile software delivery pipelines to ship new features faster, more securely and with confidence for both Linux and Windows Server apps.

What Is Docker?

Docker is the world's leading software container platform. Developers use Docker to eliminate "works on my machine" problems when collaborating on code with co-workers. Operators use Docker to run and manage apps side-by-side in isolated containers to get better compute density. Enterprises use Docker to build agile software delivery pipelines to ship new features faster, more securely and with confidence for both Linux and Windows Server apps.

What Is Docker?

Docker is the world's leading software container platform. Developers use Docker to ~~eliminate~~ "works on my machine" problems when collaborating on code with co-workers. Operators use Docker to run and manage apps side-by-side in isolated containers to get better compute density. Enterprises use Docker to build agile software delivery pipelines to ship new features faster, more securely and with confidence for both Linux and Windows Server apps.

What Is Docker?

Docker is the world's leading software container platform. Developers use Docker to eliminate "works on my machine" problems when collaborating on code with co-workers. Operators use Docker to run and manage apps side-by-side in isolated containers to get better compute density. Enterprises use Docker to build agile software delivery pipelines to ship new features faster, more securely and with confidence for both Linux and Windows Server apps.

What Is Docker?

Docker is the world's leading software container platform. Developers use Docker to eliminate "works on my machine" problems when collaborating on code with co-workers. Operators use Docker to run and manage apps side-by-side in isolated containers to get better compute density. Enterprises use Docker to build agile software delivery pipelines to ship new features faster, more securely and with confidence for both Linux and Windows Server apps.

What Is Docker?

Docker is the world's leading software container platform. Developers use Docker to eliminate "works on my machine" problems when collaborating on code with co-workers. Operators use Docker to run and manage apps side-by-side in isolated containers to get better compute density. Enterprises use Docker to build agile software delivery pipelines to ship new features faster, more securely and with confidence for both Linux and Windows Server apps.

Why the confusion?

- Docker has been moving fast, and there have been many updates.
- The result is that there are some discrepancies even in relatively new documents.
- There's also confusion about what Docker is.
- It's not a VM (although it can run (and did) run in one).

Containerization's Not New

Like many other 'new' technologies, Docker is a next step in the evolution of a number of existing technologies.

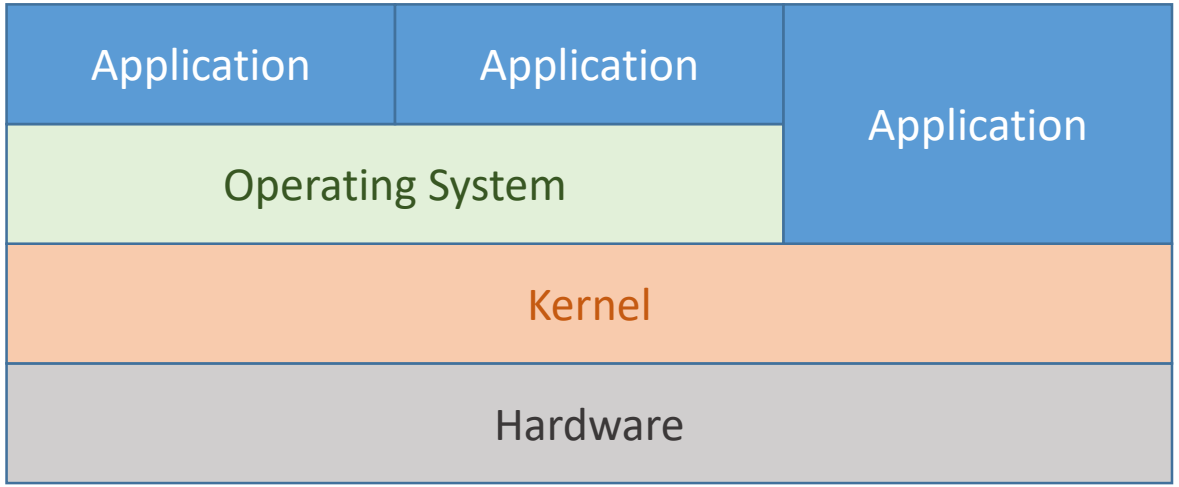
Arising from the need to prevent applications (and users) from interfering with one another on a single system.

Initially in the late 1970s, there's been steady improvements since the turn of the century. Especially with the Linux kernal.

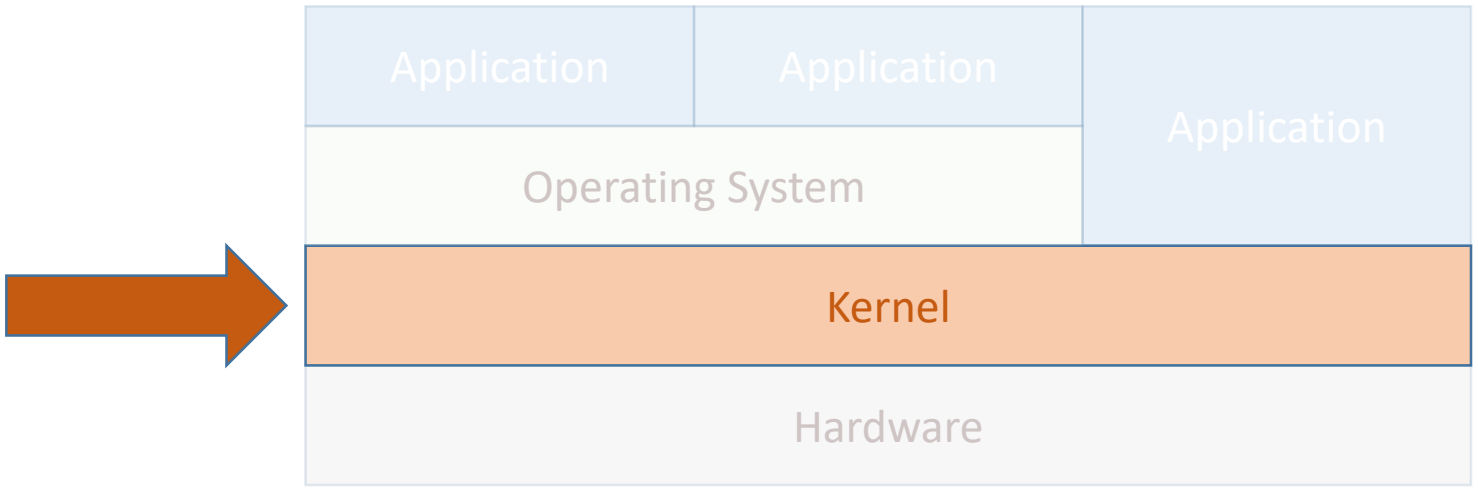
It’s not a VM!

Docker is ***not*** a virtual machine.

A simplified view of the innards of a standard computer.



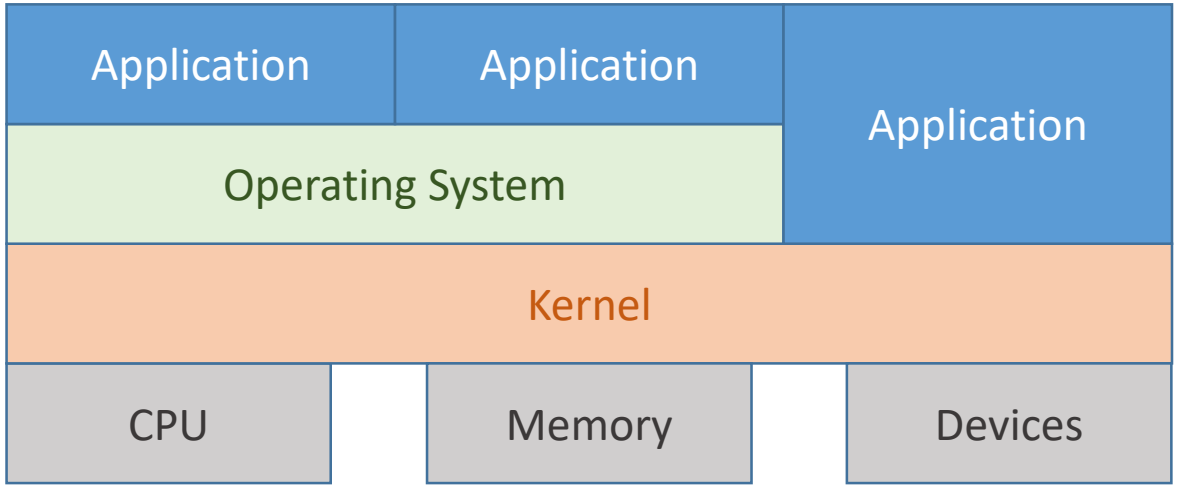
Important: what's a kernel?



Important: what's a kernel?

- The kernel is the core of the Operating System. It controls the entire system.
- It also acts as a proxy between hardware and peripherals such as keyboards, monitors, network cards, etc.
- The kernel controls things like access to hardware, certain operations, and other features.

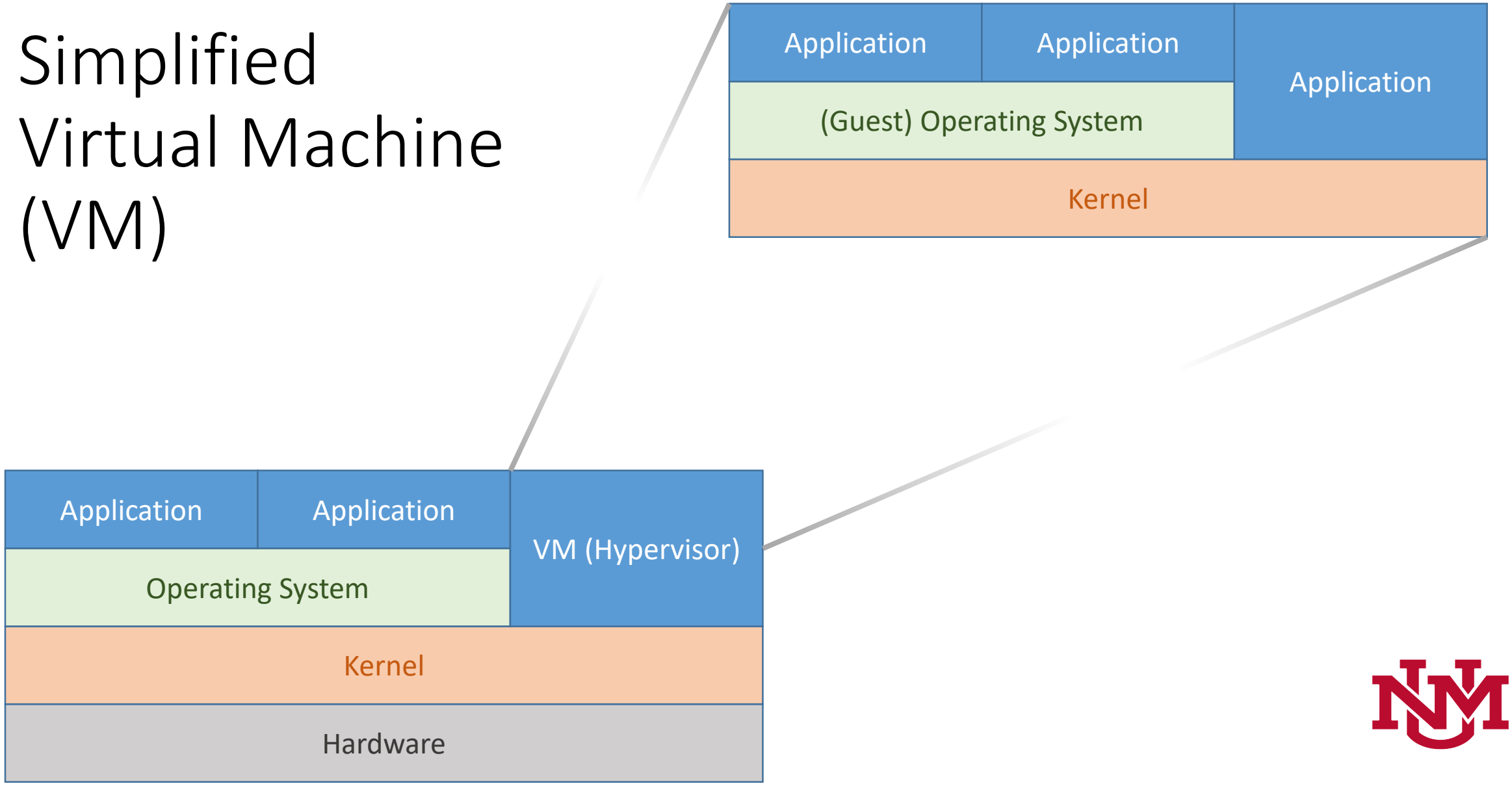
Important: what's a kernel?



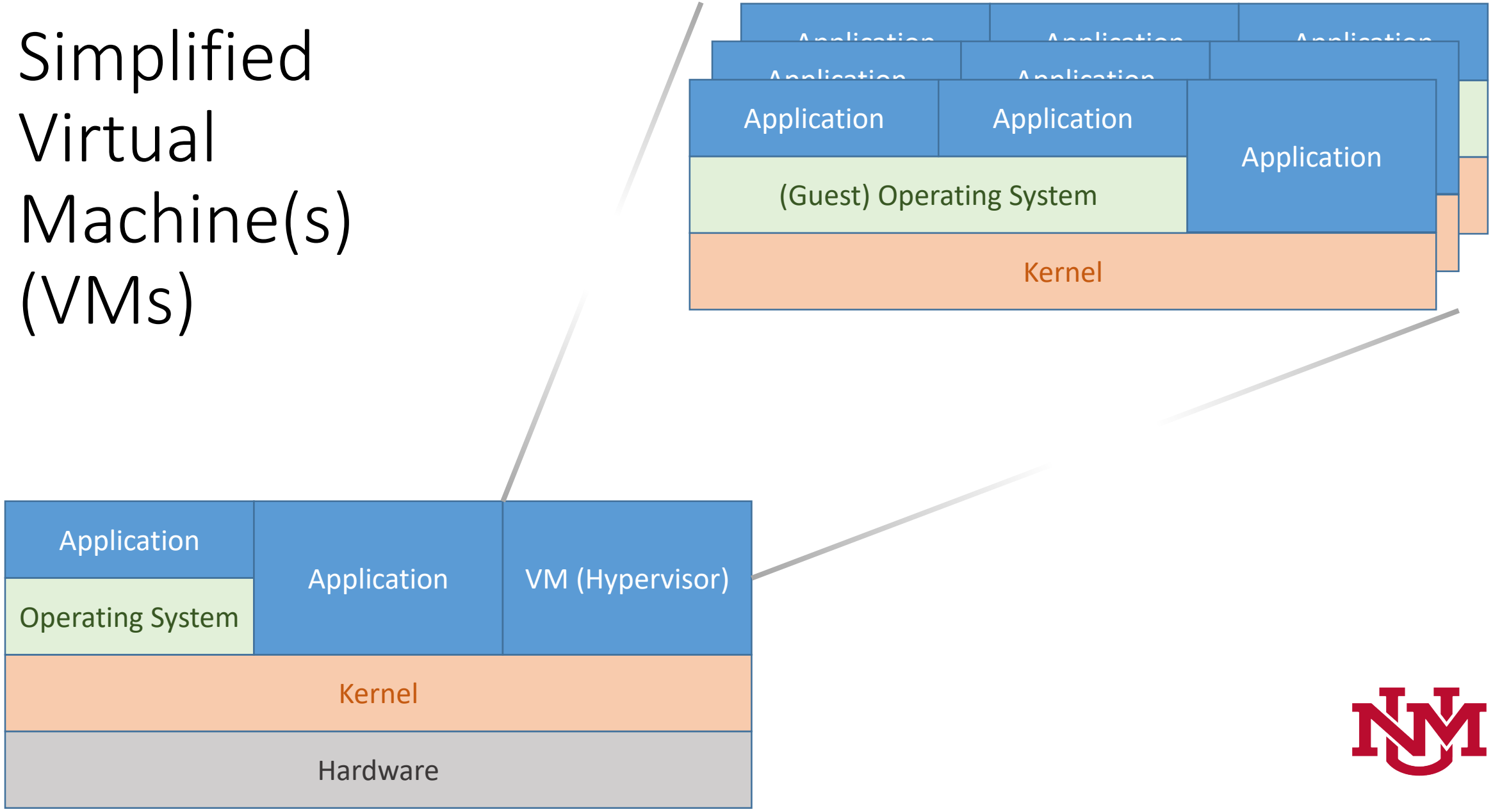
What's a Virtual Machine (VM)?

- It's a simulation of a computer, running on simulated hardware.

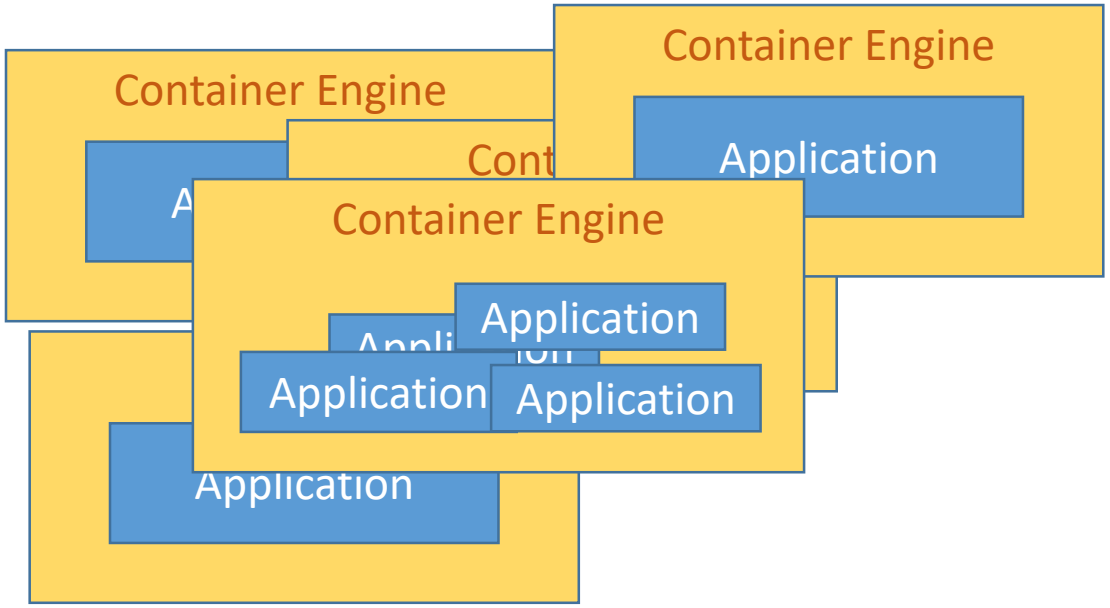
Simplified Virtual Machine (VM)

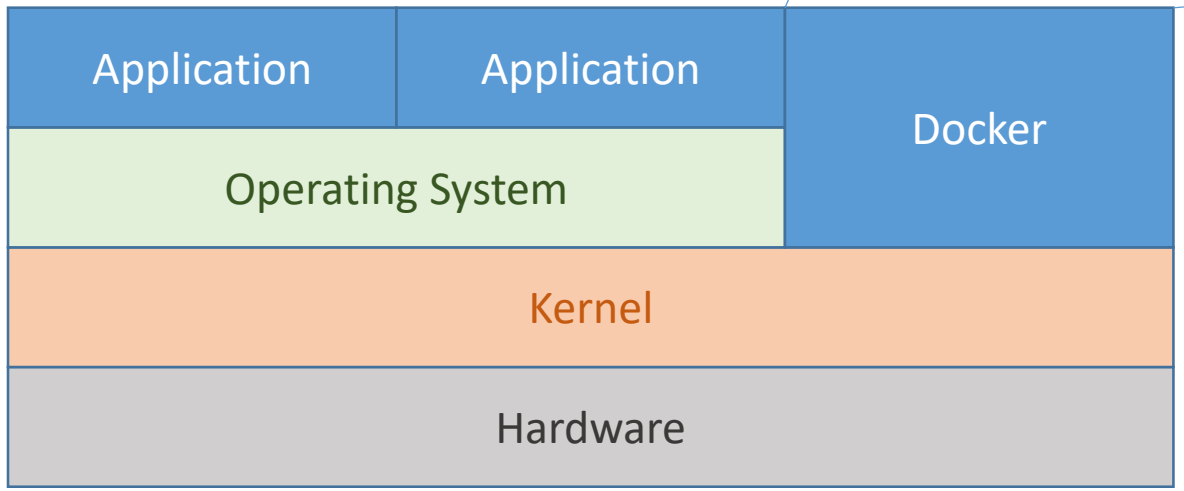


Simplified Virtual Machine(s) (VMs)

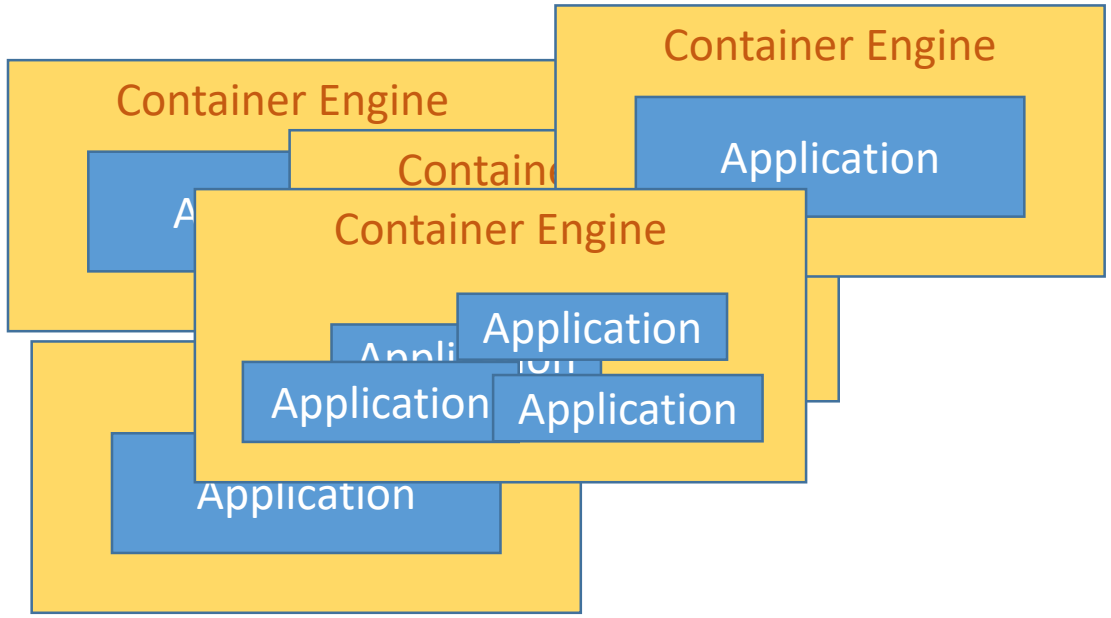


It's a Container Manager!

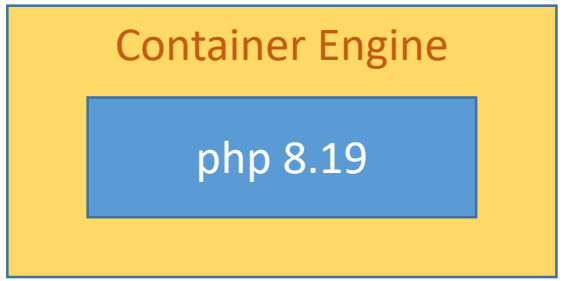


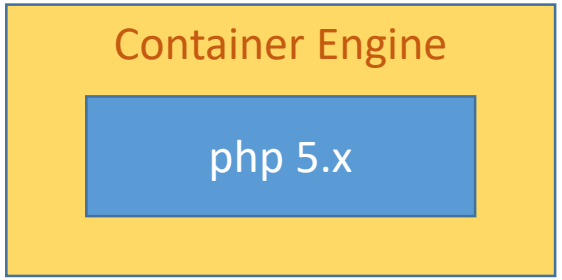


Runtime Isolation

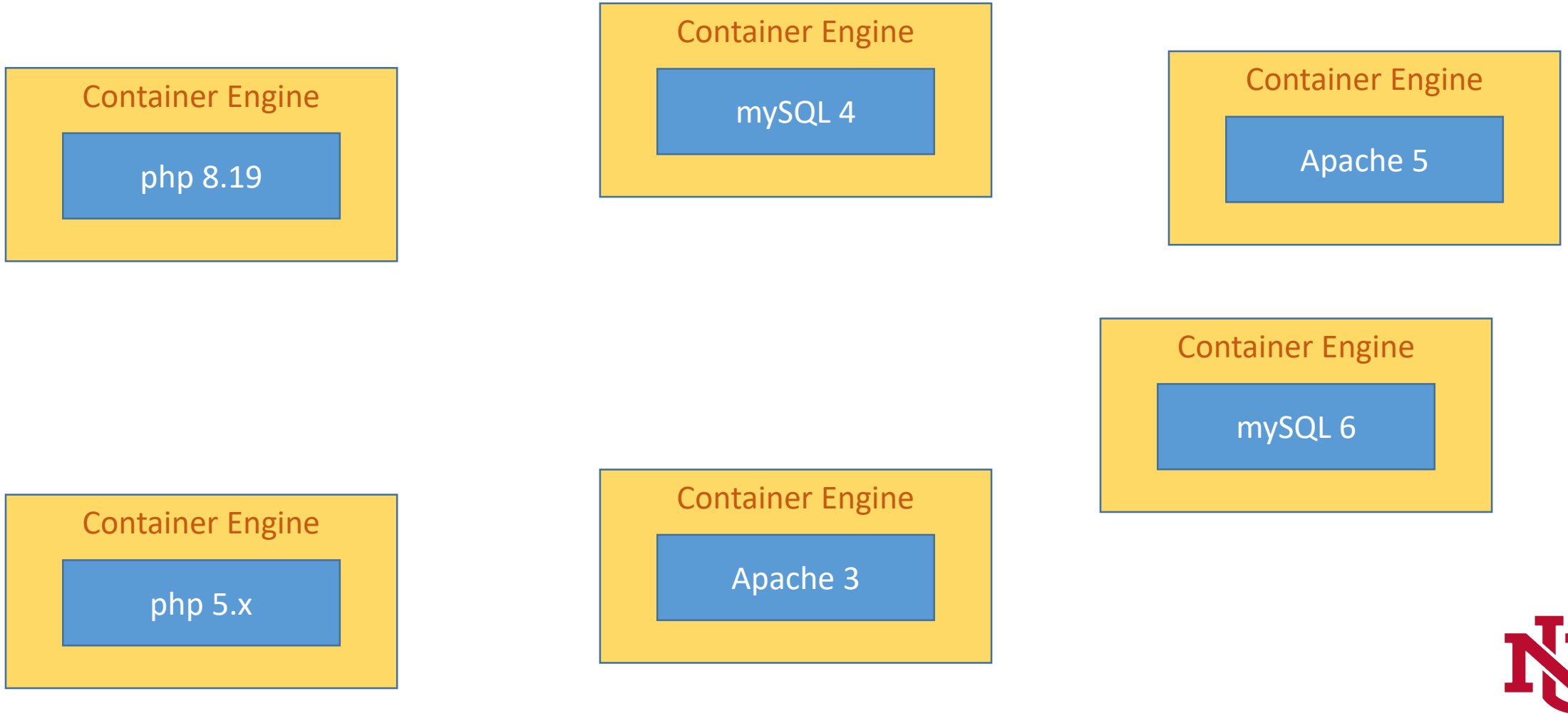


Runtime Isolation

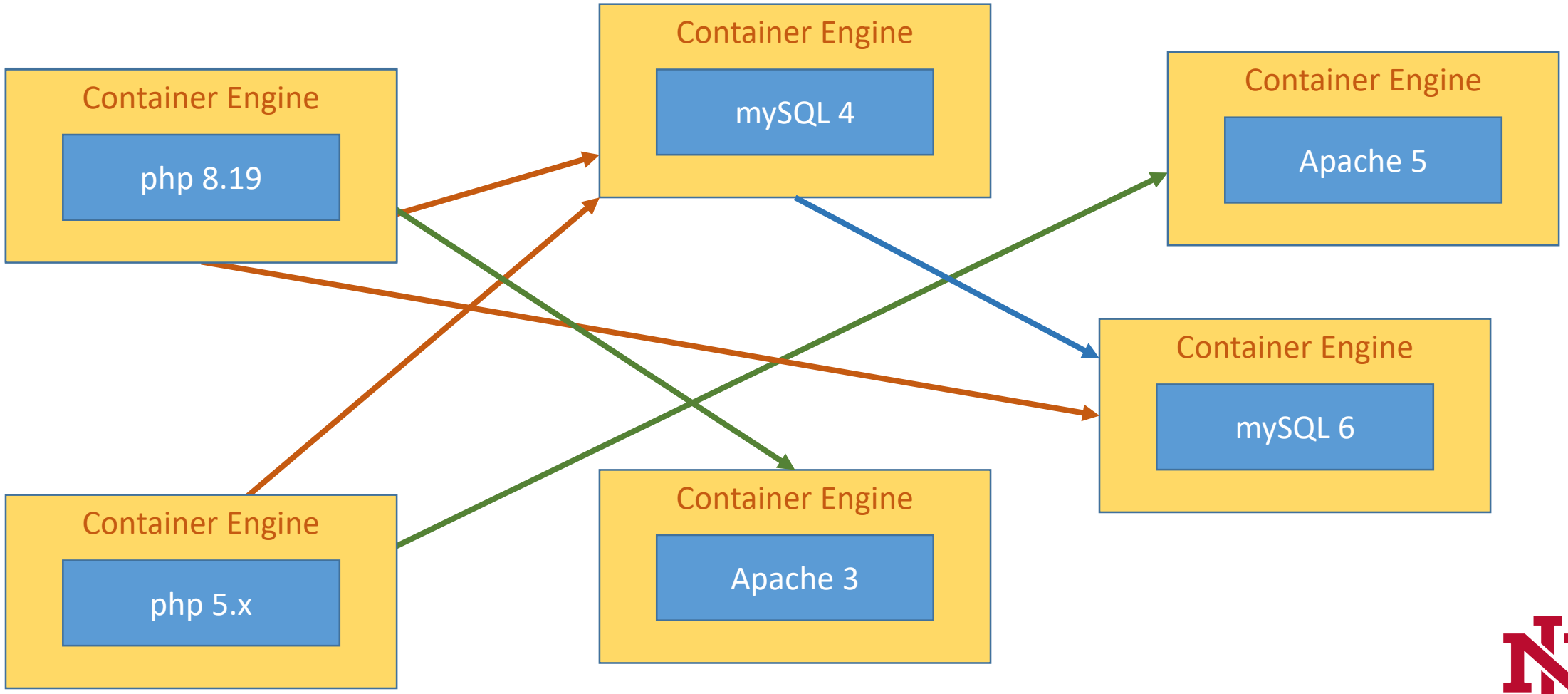




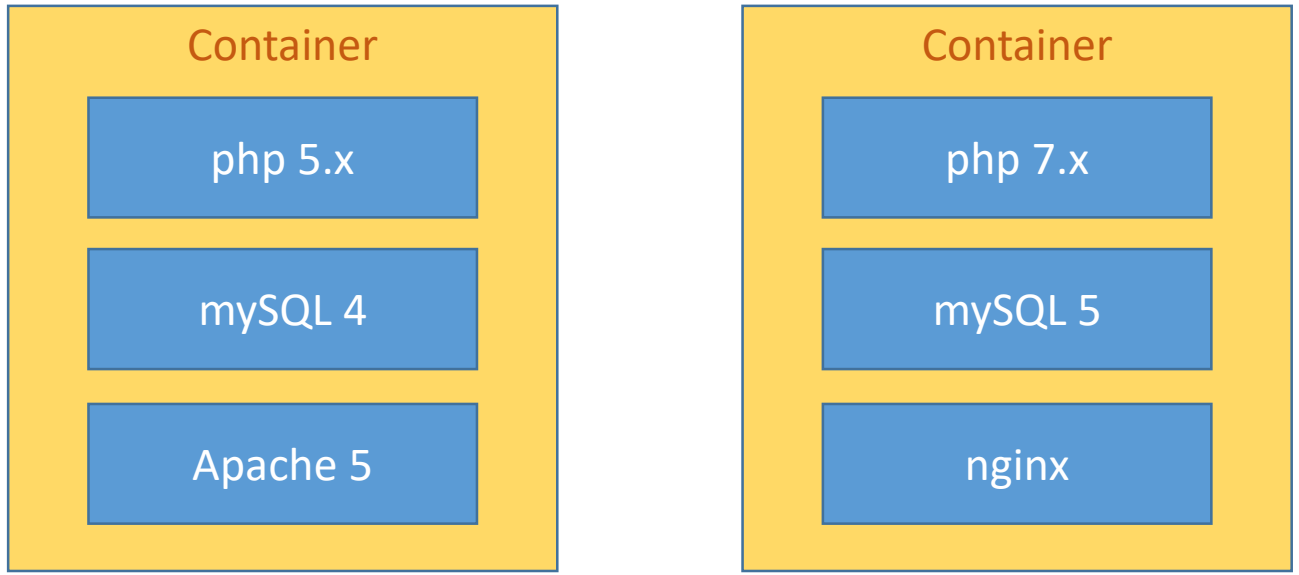
Runtime Isolation



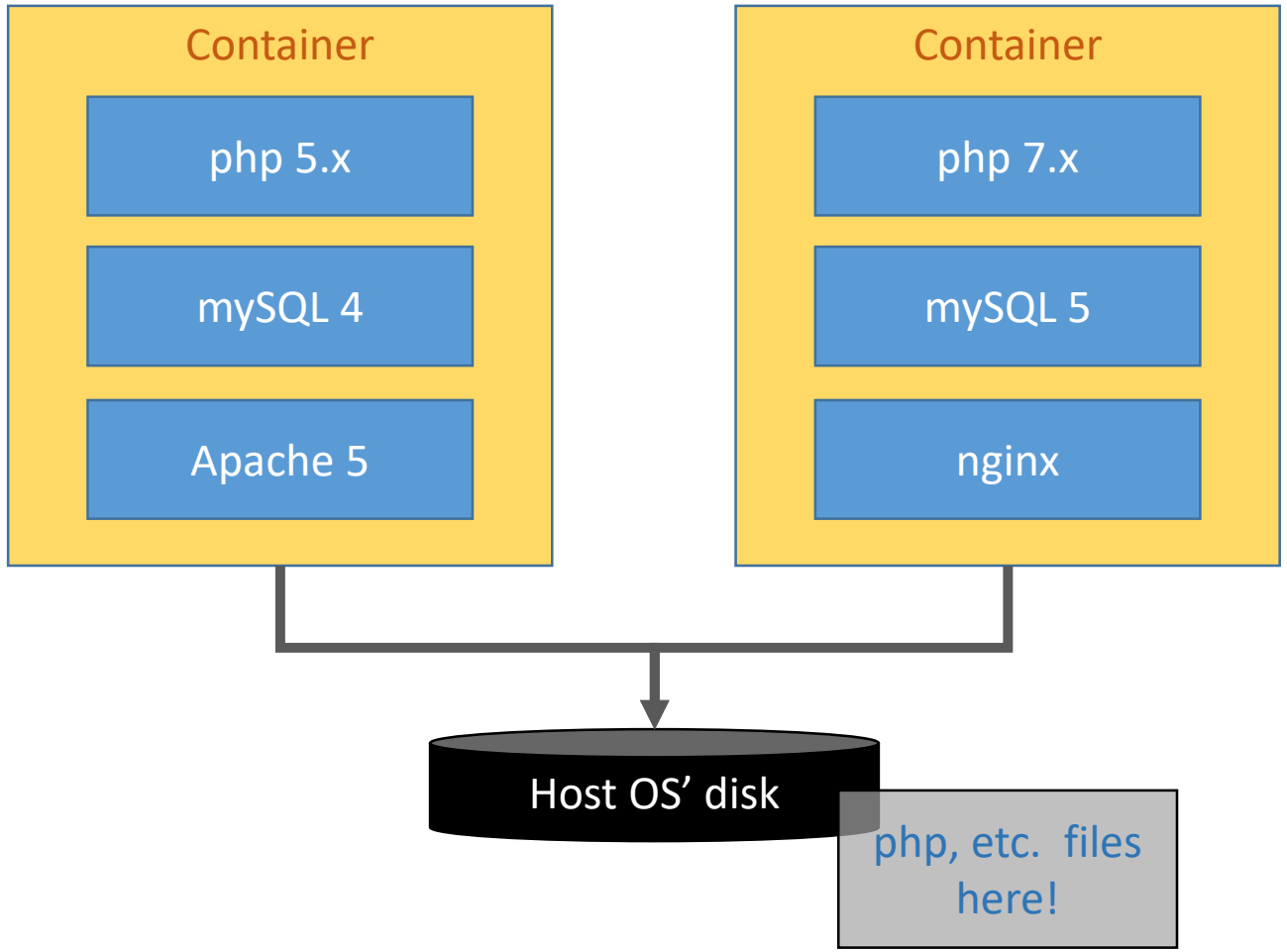
Runtime Isolation



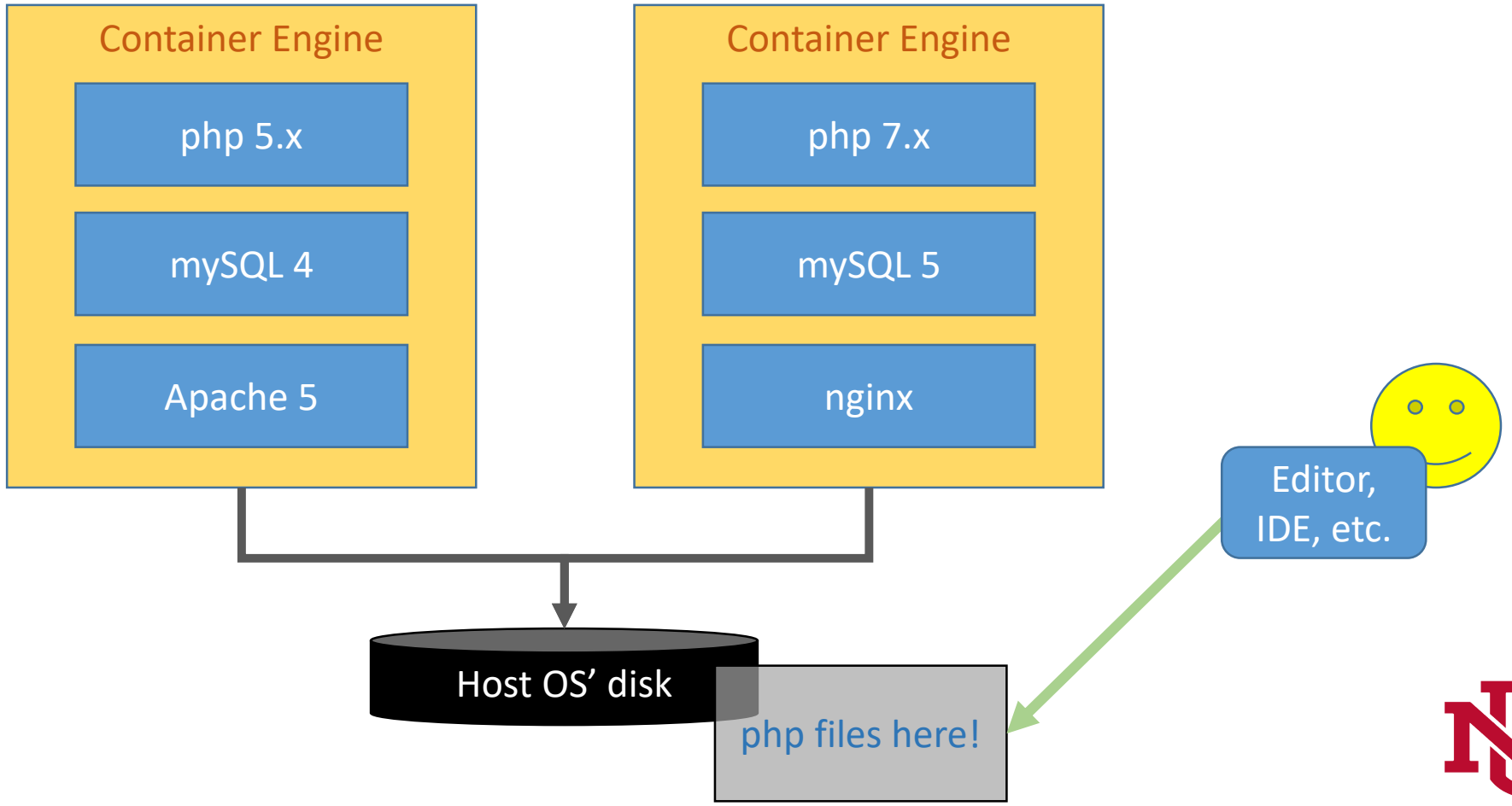
Runtime Isolation



Runtime Isolation



Runtime Isolation



What makes it so cool (good, easy)…

What makes it so cool (good, easy)…

It utilizes a single kernel, so it saves lots of resources.

Well-configured Docker configuration files make it easy to create stable, reusable development environments.

Less finicky than VMs(?).

Easier to keep up with updates to php, MySQL, etc. (Because you're not relying on your *AMP vendor to update).

Easier to segregate Projects.

What makes it so difficult…

It doesn’t make managing containerized apps any easier.

IOW, to master Docker a good working knowledge of *ix systems, networking, resources, automation, etc. is required.

Therefore, customizing Docker is more Ops than Dev.

Creating customized Images requires some knowledge of *ix (moderate to expert)

Docker Repository

- A centralized location with many Images.
- Some are official, many are from the community.
- There’s probably an existing Image that does what you need.
- Link!

Main use-cases for developers

- Develop locally and deploy files
- Develop locally and deploy Containers.

Develop locally and deploy files

- Eliminate *AMP (MAMP, XAMP, etc.)
- Easily try new versions of your stack.
  - For example, want to see how your php 5 code runs on php 7?
- Close parity with your production environment.
  - But not necessarily exact
    - Eg: Prod = php 5.6.12; your dev = php 5.6.19

Develop locally and deploy files - workflow

- Install Docker, etc.
- Get an image that suites you.
- Get your database, browser, etc., going.
- Code (implement new features, fix bugs, etc.)
  - (Edit, save, refresh).
  - Commit to your versioning tool, if you're using one.
- Deploy your code (for testing, acceptance, production, etc.)
- Does NOT eliminate works on my machine issues.

Develop locally and deploy Containers.

- Requires your Ops team to be ready (because they need to have Docker running on target systems).
- Install Docker.
- Get an image that suites you.
- Code (implement new features, fix bugs, etc.)
- Deploy your Container (for testing, acceptance, production, etc.)
- Eliminates works on my machine issues.

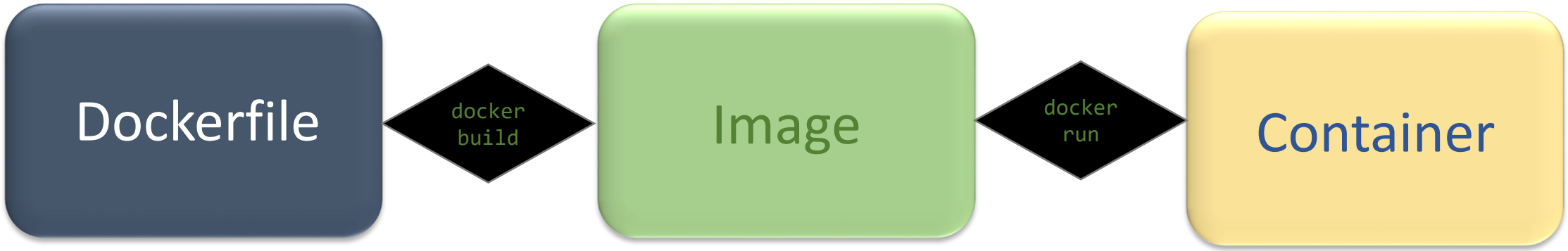
Pros

- Great deal of freedom for developers.
- Less Ops knowledge than VMs.
- Eliminate VMs.
- Eliminate *AMP stacks.
- In a fully realized Docker shop, less stressful deployments
  - Blue Green
- Widely developed, adopted and supported by very large enterprises like Google, Red Hat, Microsoft, etc., etc.

Cons

- Great deal of freedom for developers.
  - My project uses Lua, Caddy and CockroachDB!!
  - Neat, huh!?!?
- Mastery requires non-trivial knowledge of *ix.
  - More Ops than Dev.
- The new hotness
  - But it’s probably here to stay.

The Process – Innards –Essential Knowledge



A Dockerfile is a text document…

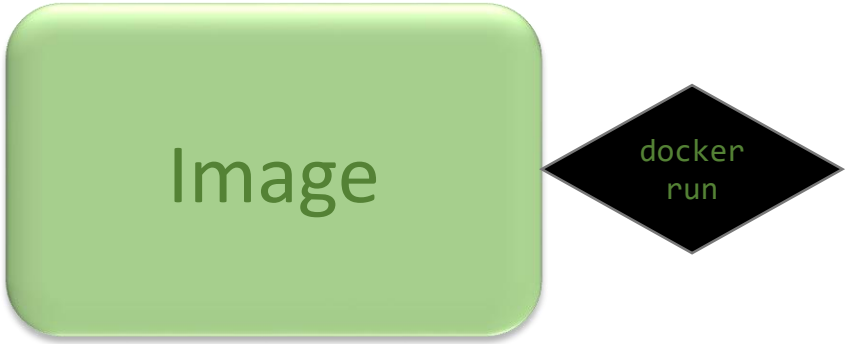
An image is a lightweight, stand-alone, executable…

A container is a runtime instance of an image…

Dockerfile

A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Using `docker build` users can create an automated build that executes several command-line instructions in succession[1].

Image



An image is a lightweight, stand-alone, executable package that includes everything needed to run a piece of software, including the code, a runtime, libraries, environment variables, and config files[2].

Containers

A container is a runtime instance of an image—what the image becomes in memory when actually executed.

It runs completely isolated from the host environment by default, only accessing host files and ports if configured to do so[2].

What about security?

- What about it?

Demo…

Sung-Hee Lee

References and Resources

- http://rhelblog.redhat.com/2015/08/28/the-history-of-containers/
- https://blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016
- https://en.wikipedia.org/wiki/Kernel_(operating_system)
- https://en.wikipedia.org/wiki/Hypervisor
- Redhat's developer site - https://developers.redhat.com/.
- Docker's site - https://www.docker.com/.
- Play with Docker - https://labs.play-with-docker.com/.