Docker ♥ Node.js

Robert Schultz, Architect, AncestryHealth

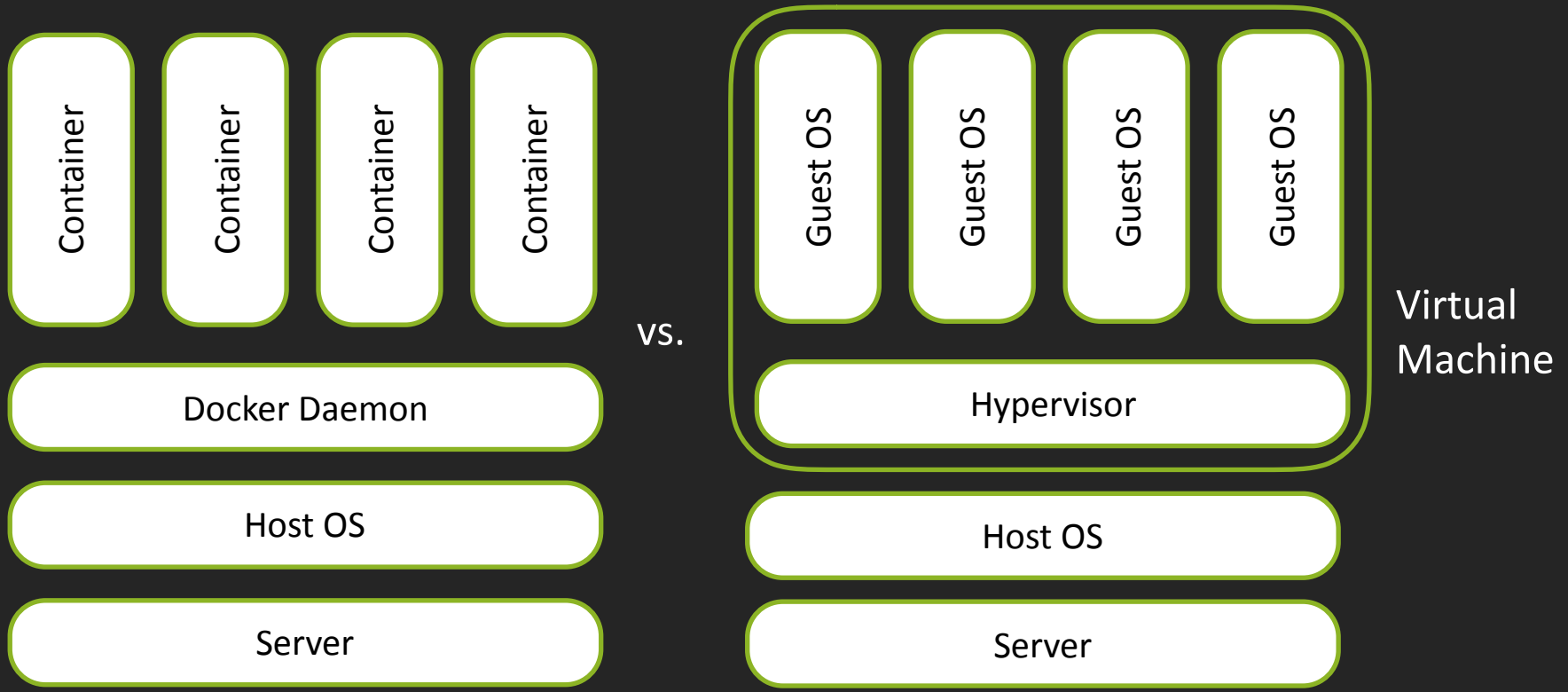
Topics for today will give you a basic understanding of Docker and how it works well with Node.js

Intro to Docker
Why Docker and Node.js
Building a Container with Node.js
Docker Machine
Docker Compose
Docker Hub
Monoliths are Bad
Microservices
Debugging

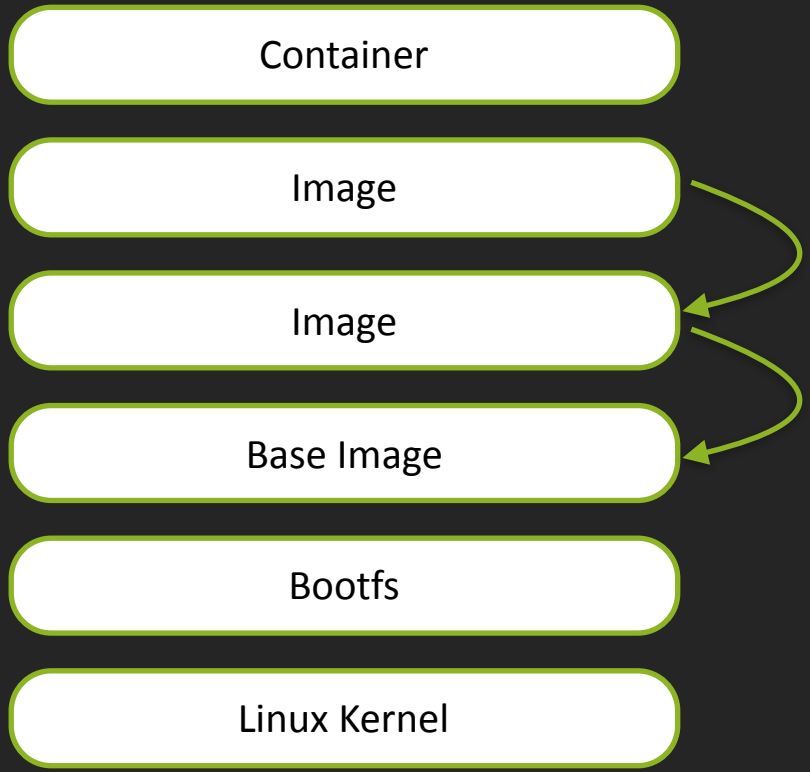
Intro to Docker

- Containers share the same operating system kernel and doesn't require a guest operating system
- Allows you to package an application with all of its dependencies into a standardized unit for software development
- Run processes in an isolated environment
- The same Docker image can be deployed to any environment
- Works off of a virtual filesystem layer
- Link between containers easily with environment variables

Containers vs. Virtual Machines



Container Architecture



Traditional Deployment

- Traditional tools for deployment includes tools such as Puppet, Chef, Salt, Ansible
- Great to automate the provisioning of the host, set environment variables, configure dependencies
- These methods take time to provision each host, can be slow or difficult to rollback

Docker Deployment

- No dedicated host needed for provisioning
- Containers start as fast as starting the process (ms)
- Application and environment is already built
- Utilization is only what your application needs to run
- Ephemeral architecture
- Very easy to scale containers up or down as needed
- Link multiple containers together
- Spin up entire architecture and dependencies for local development quickly

Docker & Node.js

- Docker is process based and Node.js is a lightweight process
- Startup time is average 50ms for Node.js
- Promotes an API driven architecture through many small services
- Node.js is great at efficient networking

Demo

- Build a simple Docker container and run it
- Demonstrate Docker commands

Linking Containers

- Built in feature to link multiple containers together
- This exposes environment variables with address and port info
- A mapping in /etc/hosts also gets automatically created

Demo

- Start up multiple containers, mapping port and linking between the two containers

Docker Machine

- Tool to install and run Docker on Mac or Windows
- Interfaces with different providers; i.e. VirtualBox, AWS
- Provision and setup multiple Docker hosts
- Start, stop, restart Docker hosts

Demo

- Create a new machine to demonstrate how to interface with a Docker host

Docker Compose

- Tool to define and run multi-container applications
- Single command to launch all containers
- Great for development, testing and staging environments

Demo

- Build a Docker Compose definition file
- Launch multiple containers from using the docker-compose command

Docker Hub

- Online registry to store Docker base images
- Support for many open source projects
- Think of it as a GitHub for prebuilt Docker images to use
- Provides hosting of private registries
- When performing CI, you can push your versioned images to use to deploy at later deployment stages

Demo

- Explore the Docker Hub registry

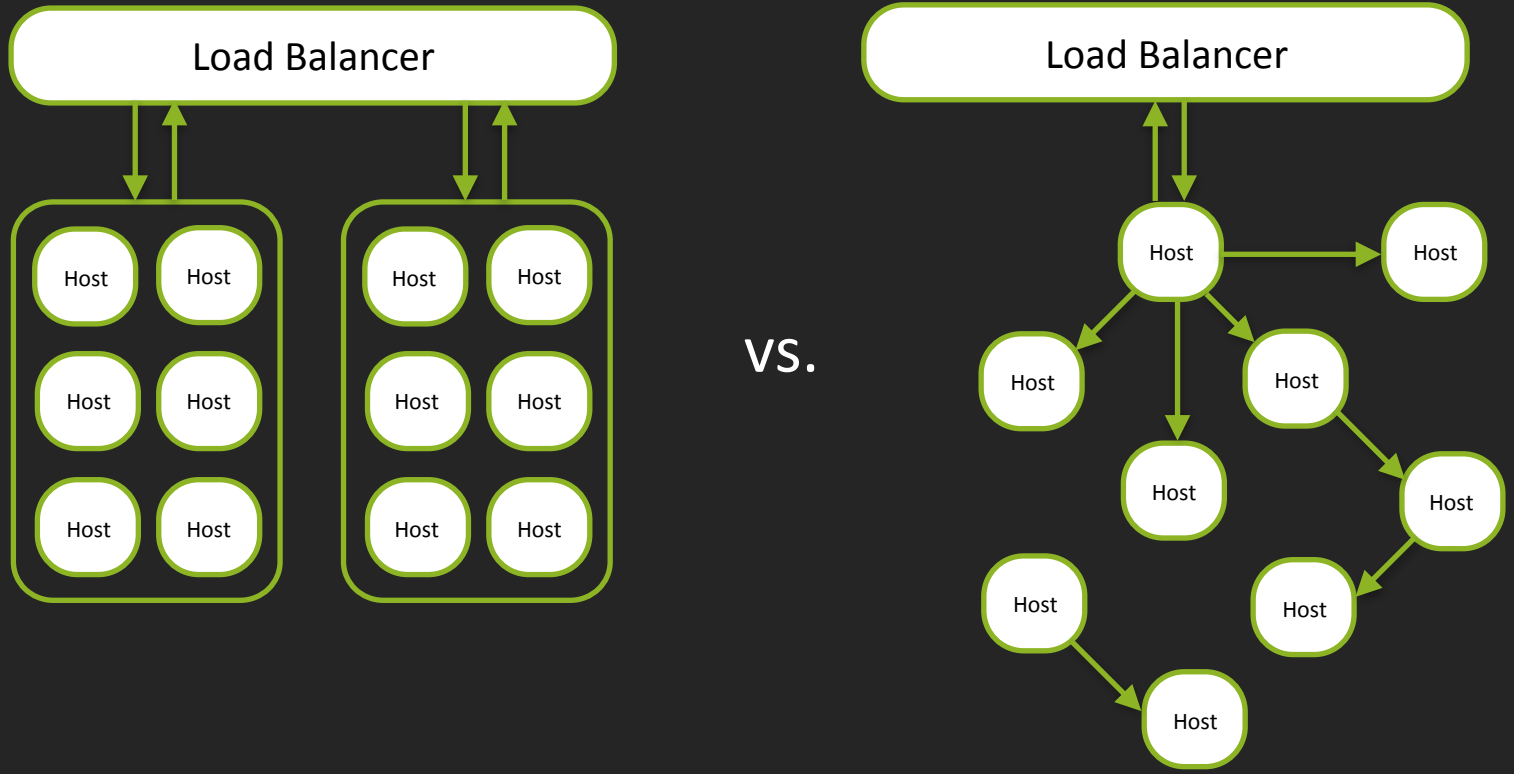
Monoliths

- Traditionally we have built big monolithic systems with everything in one process
- Monoliths come with their own set of problems we've come to identify
- They limit the scope of change in your system
- Promote tight coupling of implementation to other components
- Deploy and rollback times can become extremely long
- Deploys are more brittle and can affect other parts of the system more easily

Monoliths (cont.)

- Difficult to scale up different parts of your system
- For example, more users will be uploading photos. You should be able to scale up this component individually
- But in a monolith, you have to scale up the whole system, thus wasting resources
- Swapping out one technology for another in a monolith doesn’t happen very easily
- One thing to keep in mind, it’s sometimes best to start with one application then break as it grows

Monolith vs. Microservice



Microservices

In computing, microservices is a software architecture style, in which complex applications are composed of small, independent processes communicating with each other using language-agnostic APIs. These services are small, highly decoupled and focus on doing a small task

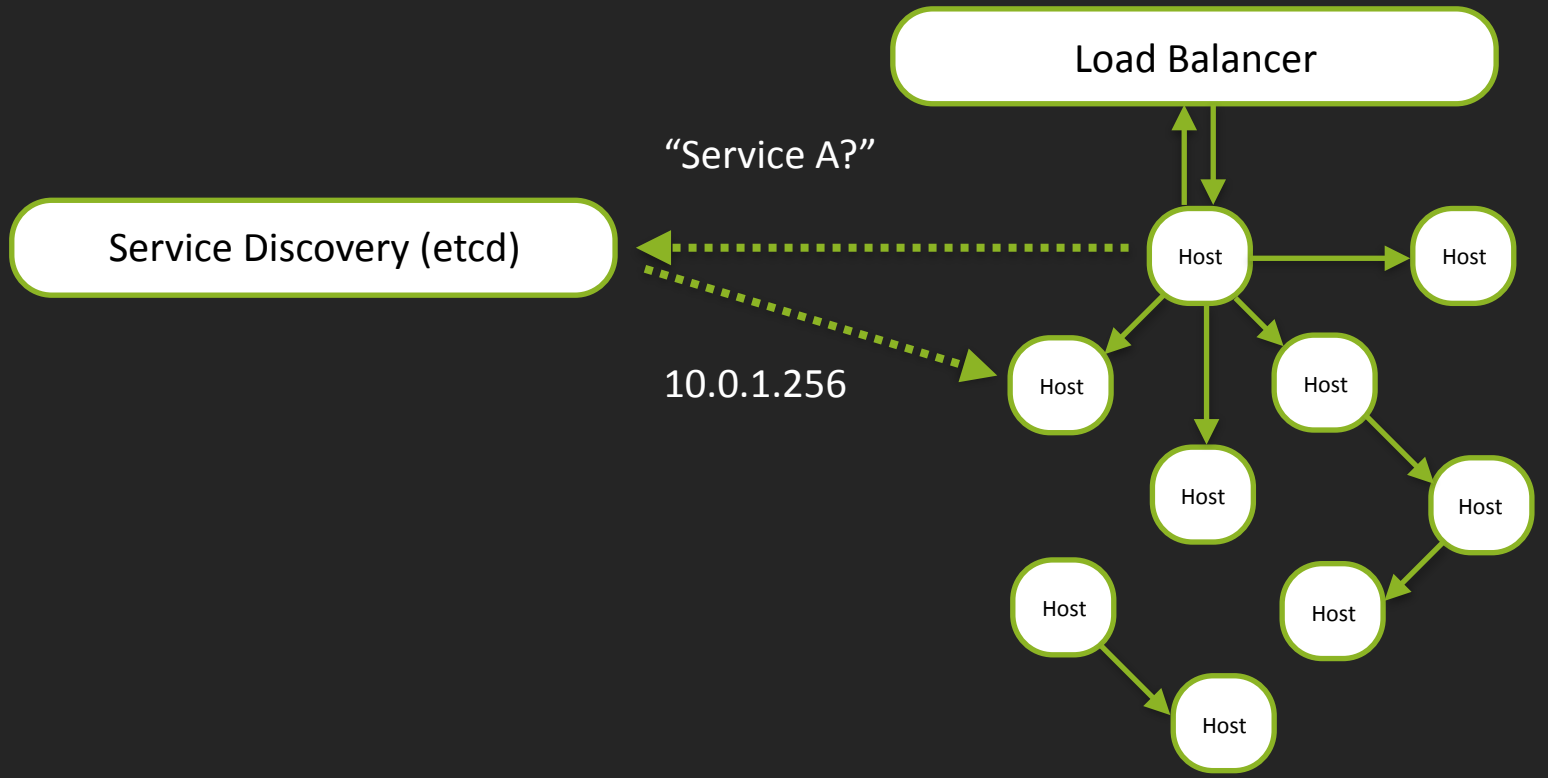
Microservices

- Docker helps enable micro services since it makes deploying an application a lot easier
- They are single purpose, again great for Docker
- When your application's code base is small, the magnitude of coupling is limited
- Loose coupling, tight cohesion
- Imagine an ecosystem of lots of single purpose containers where the application is a composition of them all

Service Discovery

- Since Docker containers should only know about their input and output, you will want to use a service discovery tool
- Provides a consistent, highly available and distributed system which defines other available services
- Dependencies can be managed outside of the application
- Manages circuit breaking when one or more services goes offline
- Consul, Zookeeper, Etcd

Monolith vs. Microservice



ancestry

Debugging

- Docker containers run a single process so debugging can be a little confusing at times
- When the process dies the container stops running
- Need to ‘exec’ into a running container
- You should never want to install or configure ssh to remotely manage your container; this violates single process rule
- Can attach to stdout and grab logs

Demo

- Start a container and debug it by using the Docker commands
- Inspect environment variables and /etc/hosts file
- Grab logs from stdout