Currently on: Configuring Docker to Communicate Over the Network

SSH commands for your VirtualBox VMs:

Ubuntu Server

*ssh -p 3023 root@127.0.0.1*

CentOS

*ssh -p 3022 root@127.0.0.1*

It is the author’s opinion that “containers are going to destroy the virtual machine model”

\*\*\*\*\*\*\*\*\*\*

Module 2

\*\*\*\*\*\*\*\*\*\*

In the past, you would have 1 app per server, which was incredibly wasteful. The server would reach on average 2-3% utilization.

Can split a physical machine into several virtual machines, each of which operates just like the original physical machine. Can then run several apps on the same server and have much better utilization.

Author describes this as “a massive improvement over the old ways”

The problem is each VM requires an OS, which adds a bunch of overhead.

The container cuts the operating system out of the equation. So instead of one app and operating system per VM, you simply have one app per container.

Technically the container does have an OS but it is incredibly lightweight.

There currently isn’t a good way to run multiple apps in one user space. The apps could potentially run over each other.

For example, assume you want to run two apps in one userspace, and each app needs its own version of a lib.so file. That’s a disaster waiting to happen. We don’t have a great way of having two different versions of the same file on the same system. This is an issue containers absolutely fix for us.

A container is an isolated instance of user space. So a system with 10 containers effectively has 10 isolated and independent instances of user space. That means we can safely install 10 apps on that system.

We need to be able to create isolated instances of things like root file systems, processing trees, networking stacks, etc. 10 containers means 10 independent, isolated views of the root file system. This means that an app running inside of container 1 can add , remove and modify files anywhere within its view of the root file system without impacting apps in other containers.

The user namespace allows us to have user accounts that have root privilege inside of a container but not outside of a container.

Through control groups (cgroups), can adjust how many system resources a specific container uses

Cgroups potentially more flexible than VCPU and VRAM on a hypervisor.

Capabilities take root privileges and break them down into smaller privileges (ex/ CAP\_AUDIT\_CONTROL, CAP\_CHOWN, etc).

===

Docker is both a company and a technology

Docker is a container runtime

Docker is an actual implementation of a container technology

Docker provides a standard runtime, meaning developers can code apps in docker containers on their laptop and literally lift them and drop them straight into docker containers on other platforms (ex/ Amazon Web Services, Azure, corporate data centers, etc). It literally doesn’t matter where, so long as the destination is running the docker runtime or daemon. It is honestly crazy how easy this is. We have never had anything like this in the enterprise before.

Docker is evolving into a platform (ex/ docker image format, docker registry with Docker Hub, etc)

The docker engine is truly cross-platform

===

Currently: “Windows apps developed on Windows docker containers will only run on Windows hosts running docker, and the same goes for Linux containers.”

Some people also run docker containers within virtual machines

Author’s opinion: The future will almost certainly see people dropping the overheads of the virtual machine and running Linux on the bare metal of the server and then just slapping containers on top of Linux. That way, we get far greater efficiencies and we lose all of the fat that exists inside VMs.

\*\*\*\*\*\*\*\*\*\*

Module 3

\*\*\*\*\*\*\*\*\*\*

Download Ubuntu Server. Use the LTS version (stands for long-term support)

VirtualBox is a hypervisor

For whatever reason, the author went to settings -> network -> adapter 2 -> Attached to: Host-only Adapter

Make sure to go to settings -> storage to select the server OS’ .iso file

User: eric

Password: 123456

In the partition disks step of the config, amount of volume group to use for guided partitioning was set to 80% (instead of 8.3GB)

# Downloading CentOs

CentOS is basically the community edition of redhat enterprise linux. You don’t need a RHEL subscription to use it.

Download the CentOs DVD ISO (not the Everything ISO)

Unfortunately the ISO is pretty large at 4GB, takes a while to download

\*\*\*\*\*\*\*\*

Module 4

\*\*\*\*\*\*\*\*

The docker client sends commands to the docker daemon, and the daemon responds. The daemon does the hard work of creating containers and constructing all the kernel namespaces, cgroups’ capabilities, etc required to instantiate a container. The client and daemon get installed as a single package, so we can run the client and the daemon on the same server if we want to, or they can talk over the network.

# Installing Docker on Ubuntu

*sudo su* allows you to switch to the root user

In Ubuntu, there’s an existing docker service, so you have to use docker.io instead. (Actually this doesn’t seem to be the case anymore)

Run

*apt-get update*

*apt-get install –y docker.io*

*service docker status* (to check if docker is running)

Docker will also be running whenever you reboot the server

Useful commands

*docker -v*

*docker version*

*docker info*

# Installing Docker on CentOS

Launch a terminal and run

*su root*

*yum install -y docker*

*systemctl start docker.service*

*systemctl status docker.service*

Useful commands

*docker -v*

*docker version*

*docker info*

---

# Granting Docker Control to Non-root Users

*docker run -it ubuntu /bin/bash*

This starts up a new container. The -it makes it interactive and assigns it a tty. Then we’re basing the container on an Ubuntu image, and inside of the container we’re going to run the bash process.

When you try to do this as a user, permission is denied.

Check groups with

*cat /etc/group*

And you will see docker:x:112 at the bottom.

Now add the user to the docker group by doing the following

*sudo gpasswd -a eric docker*

Also need to logout and log back in before docker run will work (caching issue).

Even if you’re only logged in as a user on the Ubuntu server, inside the docker container you will have root access. In the shell, you will see something like

*root@<hash>:/#*

Where the hash is the ID of the container

Use *exit* to exit and close the container.

# Configuring Docker to Communicate Over the Network

Stopping here for now (config & performance issues with VirtualBox, and the command line stuff is getting too advanced)