Docker Machine

STARTING OUR VIRTUAL MACHINES

Agenda

Intro / Prep Environments

Day 1: Docker Deep Dive

Day 2: Docker Advanced Deep Dive

Docker Machine Overview

- ❑ Docker Machine is a **tool** that lets you install Docker Engine on virtual hosts, and manage the hosts with `docker-machine` commands.
- ❑ You can use Machine to **create** Docker hosts on your local Mac or Windows box.
- ❑ Also, on your company network, in your data center, or on **cloud** providers like AWS or Digital Ocean.

Docker Machine Overview

- Using `docker-machine` **commands**, you can:
  - Start
  - Inspect
  - Stop
  - Restart a managed host
  - Upgrade the Docker client and daemon
  - Configure a Docker client to talk to your host.

Setting Machine Environment

- Point the Machine CLI at a running, managed host, and you can run `docker` commands directly on that host.
- For example, run `docker-machine env` default to point to a host called default.
  - NOTE: You may have to name the VM specifically, like this:

```
docker-machine env <name>
```

Setting Machine Environment

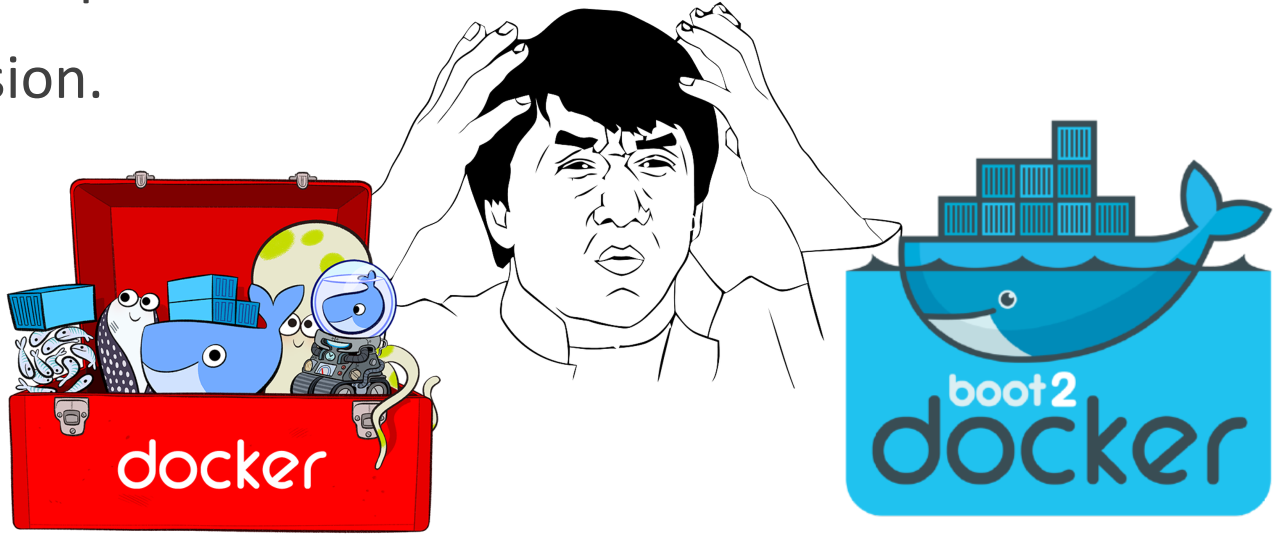
- Once you follow the on-screen instructions to complete the `env` setup, you can run `docker ps`, `docker run hello-world`, and so forth.

- Output:

```
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://XXX.XXX.XX.XXX:XXXX"
export DOCKER_CERT_PATH= <PATH>
export DOCKER_MACHINE_NAME= <NAME>
# Run this command to configure your shell:
# eval $(docker-machine env dev)
```

The Good Old Days

- Machine *was* the **only** way to run Docker on Mac or Windows previous to Docker v1.12.
- Docker for Mac & Docker for Windows are available as **native** apps *now* and are the better choice for this use case on newer desktops and laptops.
- The installers for Docker for Mac and Docker for Windows include Docker Machine, along with Docker Compose.
- So, no more need for confusion.

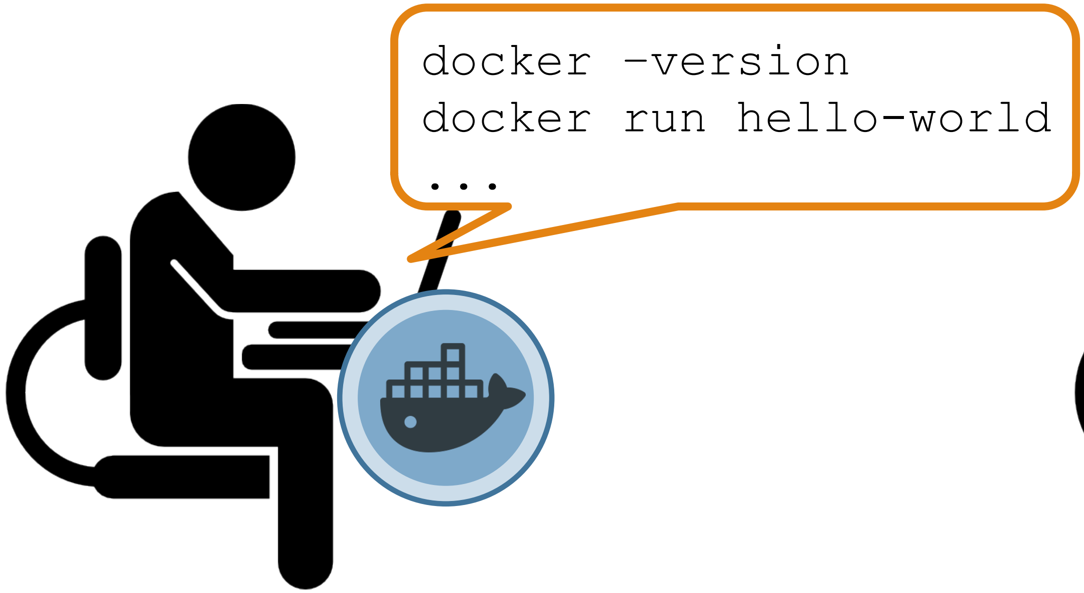


Why Use It, Now?

- ❑ Docker Machine enables you to provision multiple remote Docker hosts on various flavors of Linux.
- ❑ Additionally, Machine allows you to run Docker on older Mac or Windows systems, as described in the previous topic.
- ❑ Basically, Docker Machine has two broad use cases:

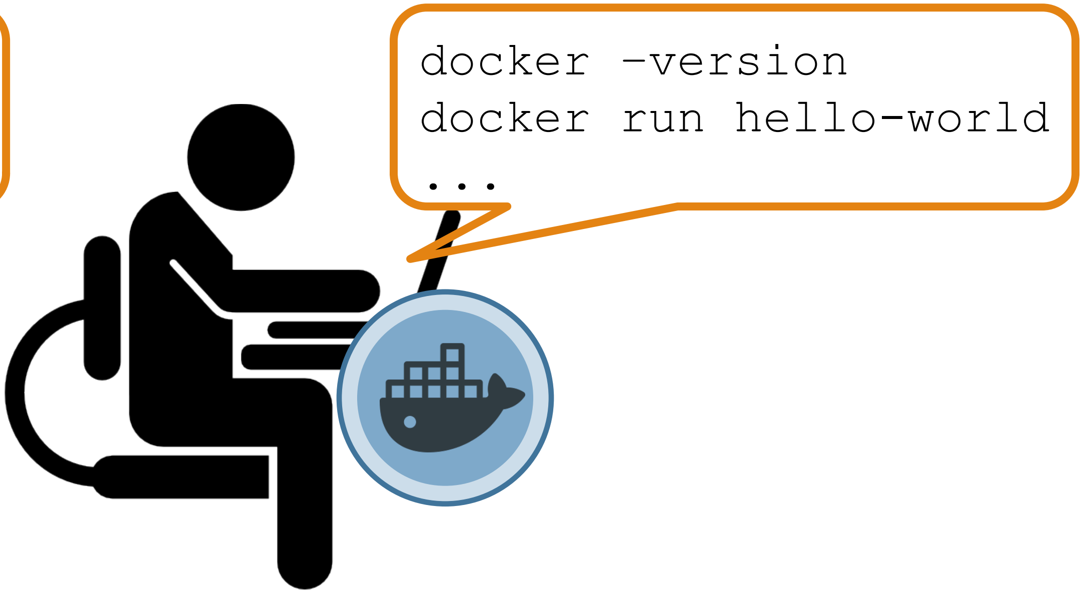
Why Use It, Now?

#1 - You own an older desktop system and want to run Docker on Mac or Windows



```
docker –version
docker run hello-world
...
```

Docker Machine for Mac



```
docker –version
docker run hello-world
...
```

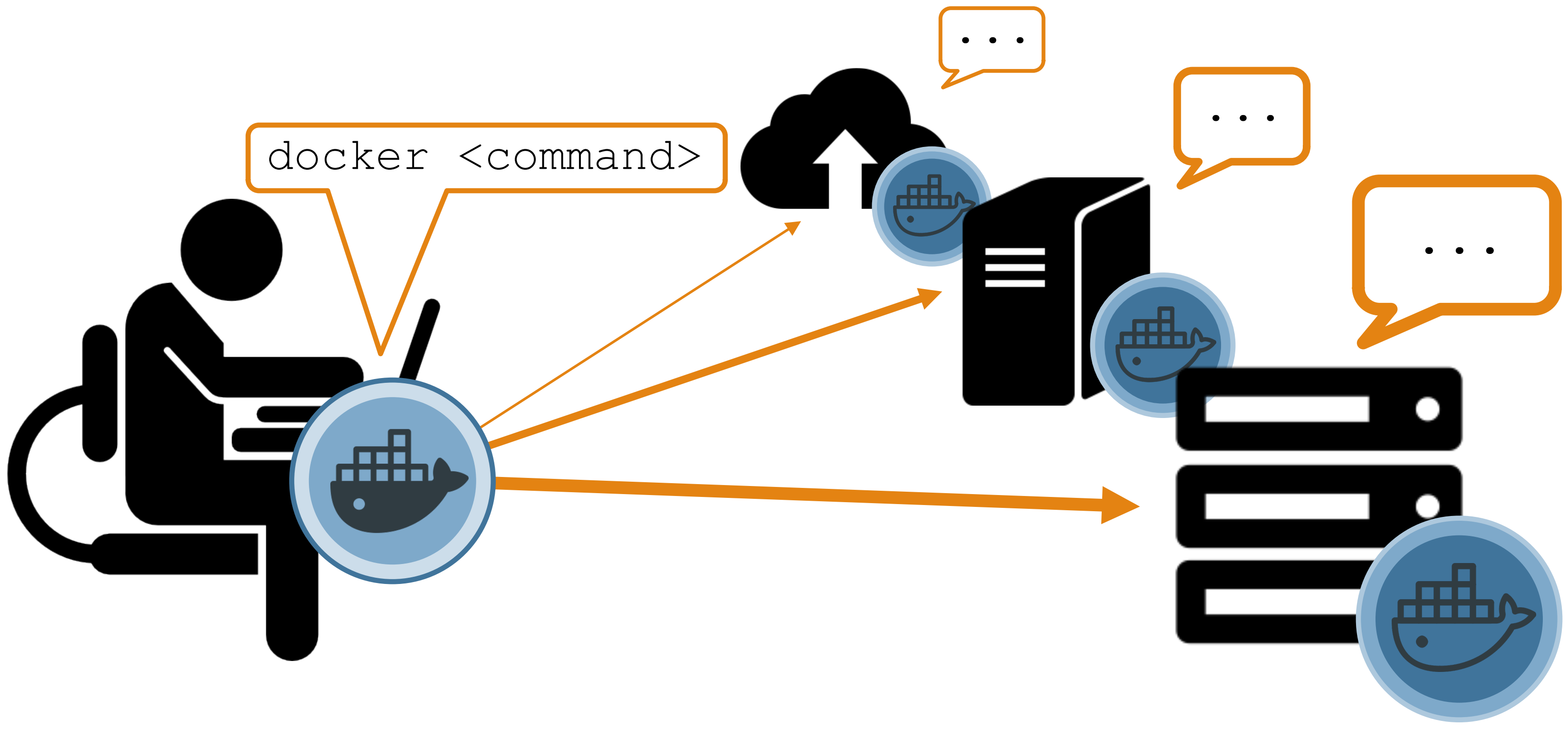
Docker Machine for Windows

The Older Desktop Option

- ❑ If the computer doesn’t meet the requirements for DockerforMac and Docker for Windows apps, then you need Docker Machine in order to run Docker Engine locally.
- ❑ Installing Docker Machine on a Mac or Windows box with the Docker Toolbox installer, provisions a local virtual machine with Docker Engine.
- ❑ This gives you the ability to connect it, and run docker commands.

Why Use It, Now?

#2 - You want to provision Docker hosts on remote systems



Remote Host Option

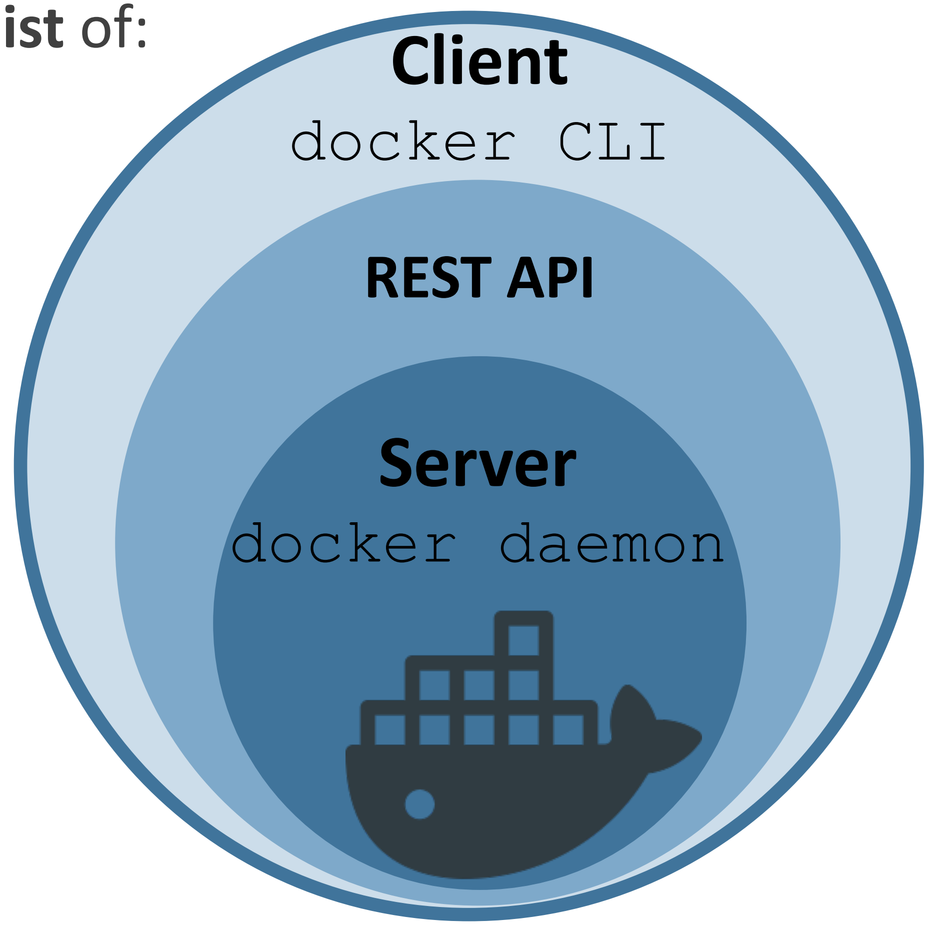
- Rememeber, Docker **Engine** runs natively on Linux systems.
- If you have a Linux box as your **primary** system, and want to run Docker commands, all you need to do is download and install Docker Engine.
- However, if you want an efficient way to provision multiple Docker hosts on a network, in the cloud or even locally, you **need** Docker Machine.

Remote Host Option

- ❑ Whether your primary system is Mac, Windows, or Linux, you can install Docker Machine on it and use `docker-machine` commands to provision and **manage** large numbers of Docker hosts.
- ❑ Machine automatically creates hosts, installs Docker Engine on them, then configures the Docker clients. Each managed host (*machine*) is the **combination** of a Docker host and a configured client.

Docker Engine vs. Docker Machine

- When people say "*Docker*" they typically mean Docker **Engine**
- Engine, the client-server application **consist** of:
  - the Docker **daemon**
  - a REST **API** that specifies interfaces for interacting with the daemon
  - a command line interface (**CLI**) client that talks to the daemon (through the REST API wrapper)



Docker Engine vs. **Docker Machine**

- ❑ Docker Engine accepts `docker` commands from the CLI, such as `docker run <image>`, `docker ps` to list running containers, `docker images` to list images, and so on
- ❑ **Docker Machine** is a tool for provisioning and managing your Dockerized hosts (hosts with Docker Engine on them).
- ❑ Typically, you install Docker Machine on your local system. Docker Machine has its own command line client docker-machine
- ❑ You can use Machine to install Docker Engine on one or more virtual systems. These virtual systems can be local (as when you use Machine to install and run Docker Engine in VirtualBox on Mac or Windows) or remote (as when you use Machine to provision Dockerized hosts on cloud providers). The Dockerized hosts themselves can be thought of, and are sometimes referred to as, managed "***machines***".

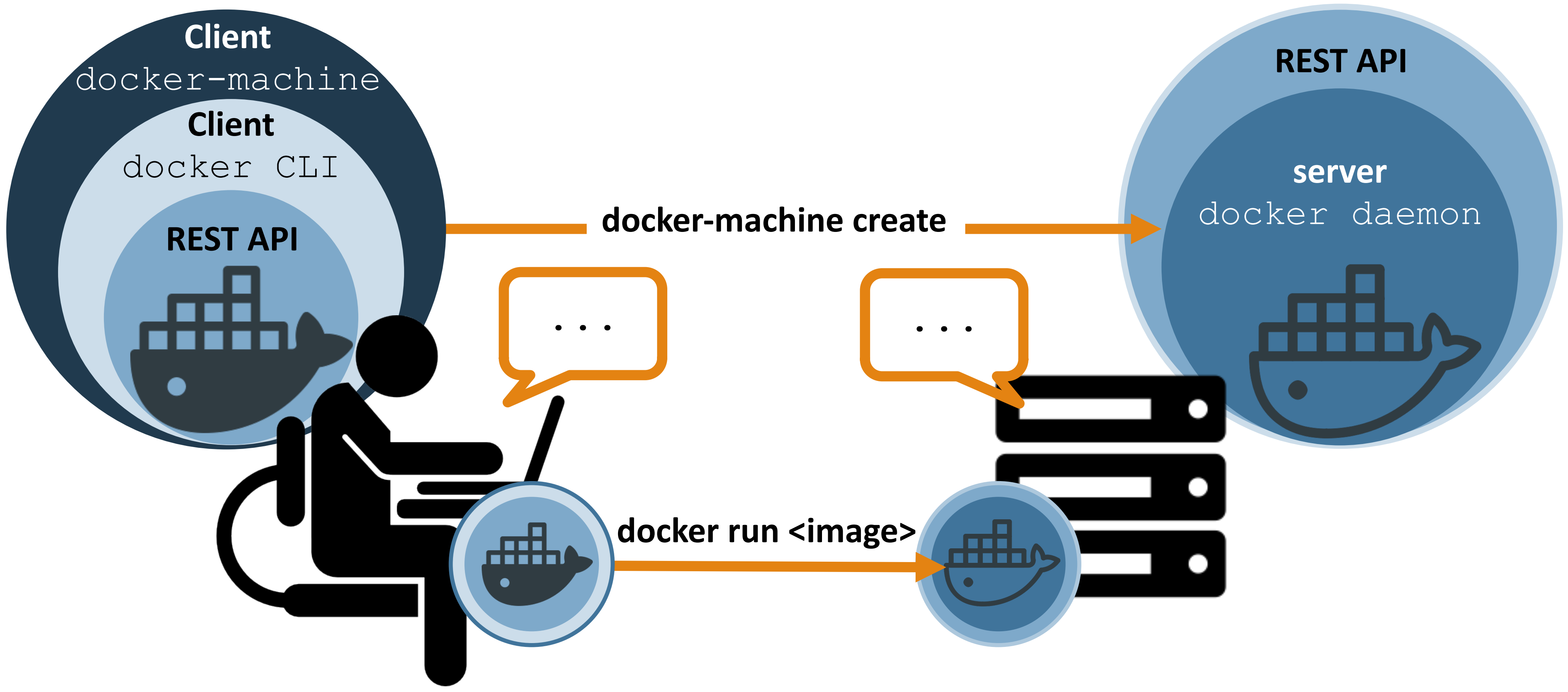
Docker Engine vs. **Docker Machine**

- Docker Engine accepts `docker` commands from the CLI, such as `docker run <image>`, `docker ps` to list running containers, `docker images` to list images, and so on
- **Docker Machine** is a tool for provisioning and managing your Dockerized hosts (hosts with Docker Engine on them).
- Typically, you install Docker Machine on your local system. Docker Machine has its own command line client docker-machine

Docker Engine vs. **Docker Machine**

- ❑ You can use Machine to install Docker Engine on one or more virtual systems.
- ❑ Virtual systems can be local (as when you use Machine to install and run Docker Engine in VirtualBox on Mac or Windows)
- ❑ Or, they can be remote (as when you use Machine to provision Dockerized hosts on cloud providers).
- ❑ The Dockerized hosts themselves can be thought of, and are sometimes referred to as, managed "***machines***".

Remote Machine Host Diagram



Getting Started Using Local VM

- Let’s take a look at using `docker-machine` to create, use and manage a Docker host inside of a **local** virtual machine.

Docker Machine Rust

- With the advent of Docker for Mac and Docker for Windows as replacements for Docker Toolbox, we recommend that you use these for your primary Docker workflows.
- You can use these applications to run Docker natively on your local system without using Docker Machine at all.

Docker Machine Rust

- However, if you want to create *multiple* local machines, you still need Docker Machine to create and manage machines for multi-node experimentation.
- Both Docker for Mac and Docker for Windows include the newest version of Docker Machine, so when you install either of these, you get `docker-machine`.

Prerequisite Information

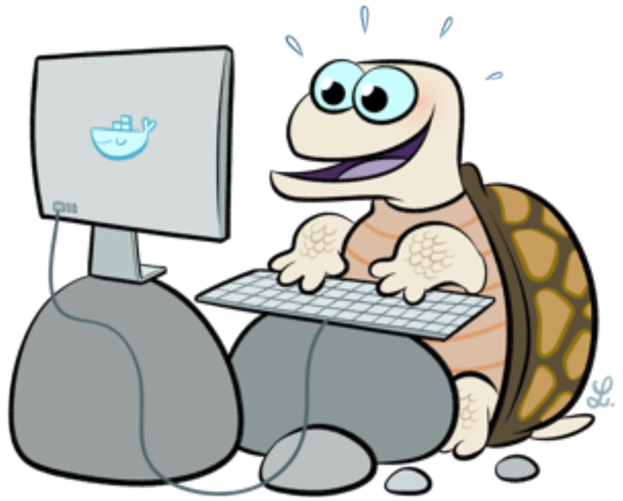
- Keep the following considerations in mind when using Machine to create local VMs.
  - **Docker for Windows** - You can use `docker-machine create` with the `hyperv` driver to create additional local machines.
  - **Docker for Mac** - You can use `docker-machine create` with the `virtualbox` driver to create additional local machines.

Windows Prerequisite Information

- ❑ **Docker for Windows**
  - ❑ Docker for Windows uses Microsoft Hyper-V for virtualization, and Hyper-V is not compatible with Oracle VirtualBox.
  - ❑ Therefore, you cannot run the two solutions simultaneously.
  - ❑ But you can still use `docker-machine` to create more local VMs by using the Microsoft Hyper-V driver.

Mac Prerequisite Information

- **Docker for Mac**
  - Docker for Mac uses HyperKit, a lightweight macOS virtualization solution built on top of the Hypervisor.framework in macOS 10.10 Yosemite and higher.
  - Currently, there is no `docker-machine create` driver for `HyperKit`, so you will use `virtualbox` driver to create local machines.
  - You can run both HyperKit and Oracle VirtualBox on the same system.



Running Containers

- To run a Docker container, you:
  - Create a new (or start an existing) Docker virtual machine.
  - Switch your environment to your new VM.
  - Use the docker client to create, load, and manage containers.
- Once you create a machine, you can reuse it as often as you like.
- Like any VirtualBox VM, it maintains its configuration between uses.

- The examples here show how to create and start a machine, run Docker commands, and work with containers.

Machine Containers Example

- Let’s look at how to create a new container, using Docker Machine.
- First, open a command `shell` or terminal window.
- Use `docker-machine ls` to list available machines:

```
docker-machine ls
```

- Output (No machines have been created):

```
NAME   ACTIVE   DRIVER   STATE   URL   SWARM   DOCKER   ERRORS
```

Machine Containers Example

- To create a Machine, we'd need to start by following the steps below:
  - **Run** `docker-machine create <command>`
  - **Pass** the appropriate driver to the `--driver` flag and provide a machine name.
  - If this is your first machine, **name** it `default` as shown in the example.
  - If you already have a "default" machine, choose **another** name for this new machine.
  - If you are using Docker for **Mac**, use `virtualbox` as the driver, as shown in this example.
  - On Docker for **Windows** systems that support Hyper-V, use the `hyperv` driver

Creating a New Machine

- So, our command should look something like this:

```
docker-machine create --driver virtualbox default
```

- Output:

```
Running pre-create checks...
Creating machine...
...
```

- This downloads a lightweight Linux distribution boot2docker, with the Docker daemon installed, pluscreates and starts a VirtualBox.

Print Newly Created Machine

- To list available machines, we want to use `docker-machine ls` again.
- This time we should find our container listed.

- Output:

```
NAME     ACTIVE DRIVER     STATE   URL ...
default  *      virtualbox Running tcp://XXX.XXX.XX.XXX:XXXX ...
```

Get Docker Machine Env Varibles

- Let’s look at the environment variable for your new VM.
- To do that, we will use the following command:

```
docker-machine env <machine-name>
```

- Output:

```
export DOCKER_TLS_VERIFY=“..."
export DOCKER_HOST=“..."
export DOCKER_CERT_PATH=“..."
export DOCKER_MACHINE_NAME=“..."
# Run this command to configure your shell:
# eval "$(docker-machine env default)"
```

Connecting Env Settings

- Now, we need to connect our shell to the new machine.
- We'll use the following command, which can be found in the output:

```
eval "$(docker-machine env default)"
```

- Okay! Now, our environment variables are set for the current shell that the Docker client will read which specify the TLS settings.
- You need to do this each time you open a new shell or restart your machine.
- You can now run Docker commands on this host.

Experiment With Machine Containers

- Run a container with `docker run` to verify your set up.
- Use `docker run` to download and run `busybox` with a simple `echo` command, like this:

```
docker run busybox echo hello world
```

- Output:

```
Unable to find image 'busybox' locally
Pulling repository busybox
e72ac664f4f0: Download complete
511136ea3c5a: Download complete
df7546f9f060: Download complete
e433a6c5b276: Download complete
hello world
```

Introducing Machine Cloud Drivers

- When you install Docker Machine, you get a set of drivers for various cloud providers.
- Also available, Docker Machine driver plugins for use with other cloud platforms are available from 3rd party contributors.
  - NOTE: These are use-at-your-own-risk plugins, not maintained by or formally associated with Docker.

Windows Azure

DigitalOcean

amazon web services™

Exploring the Cloud

- Docker Machine driver plugins are available for many cloud platforms, so you can use Machine to provision cloud hosts.
- When you use Docker Machine for provisioning, you create cloud hosts with Docker Engine installed on them.
- We’ll need to create an account with the cloud provider.

Exploring the Cloud

- We will provide account verification, security credentials, and configuration options for the providers as flags to `docker-machine create`.
- The flags are unique for each cloud-specific driver. For instance, to pass a Digital Ocean access token you use the `--digitalocean-access-token` flag.
- Let’s look at some examples on the next slide...

Digital Ocean Example

- For Digital Ocean, this command creates a Droplet (cloud host) called “docker-sandbox”:

```
docker-machine create --driver digitalocean
--digitalocean-access-token xxx docker-sandbox
```

Amazon Web Services (AWS) Example

- For AWS, this command creates an instance called “aws-sandbox” in EC2:

```
 docker-machine create --driver amazonec2 --amazonec2-access-key
AKI******* --amazonec2-secret-key 8T93C******* aws-sandbox
```

Docker Machine Swarm

- Docker Machine can also provision Docker Swarm clusters.
- This can be used with any driver and will be secured with TLS.

Start and Stop

- If you are finished using a host for the time being, you can stop it with `docker-machine stop` and later start it again with `docker-machine start`.
- Stop the machine, like this:

```
docker-machine stop <machine-name>
```

- And we would start it back up, like this:

```
docker-machine start <machine-name>
```

Lecture Summary

- ❑ In the lesson we covered Docker Machine and how to use it, generally.
- ❑ We first looked at an overview of Docker Machine and why it’s used.
- ❑ Then, we looked at how to set our machine environment variables.
- ❑ We looked at the inside functionality of Docker Machine and how it works in local and remote cases.
- ❑ Also, we compared Docker Machine and Docker Engine.
- ❑ From there, we looked at how we would create a new machine and start it.
- ❑ We then looked at how Docker Machine can function in the cloud.
- ❑ Finally, we concluded with learning how to start and stop our machines.

Lab

End of Chapter