Welcome to Week 1

Virtual Mentored Academy

Docker Fundamentals

DevelopIntelligence

A PLURALSIGHT COMPANY

Hello

HELLO
my name is
Allen Sanders
Senior Technology Instructor
Pluralsight ELS

About me...

- 27+ years in the industry
- 23+ years in teaching
- Certified Cloud architect
- Passionate about learning
- Also, passionate about Reese's Cups!

Agenda

- Containerization as a Deployment Strategy
- Docker as a Containerization Platform
- Azure Container Registry (ACR)
- Azure Container Instances (ACI)
- Docker Compose

How we’re going to work together

- Slides and words to highlight key concepts
- Demos to bring those concepts “to life”
- Lab work (which will take place in sandboxes provided by “A Cloud Guru”) for hands-on reinforcement
- NOTE: I welcome being interrupted – if you need more info, or clarification, or anything else, just break in and ask. I am here to help you.

Containerization as a Deployment Strategy

What Are the Hosting Options with Cloud?

- ❑ IaaS
- ❑ PaaS
- ❑ Serverless / FaaS
- ❑ SaaS
- ❑ Containers

What do they all mean?

6

So, What Are Containers?

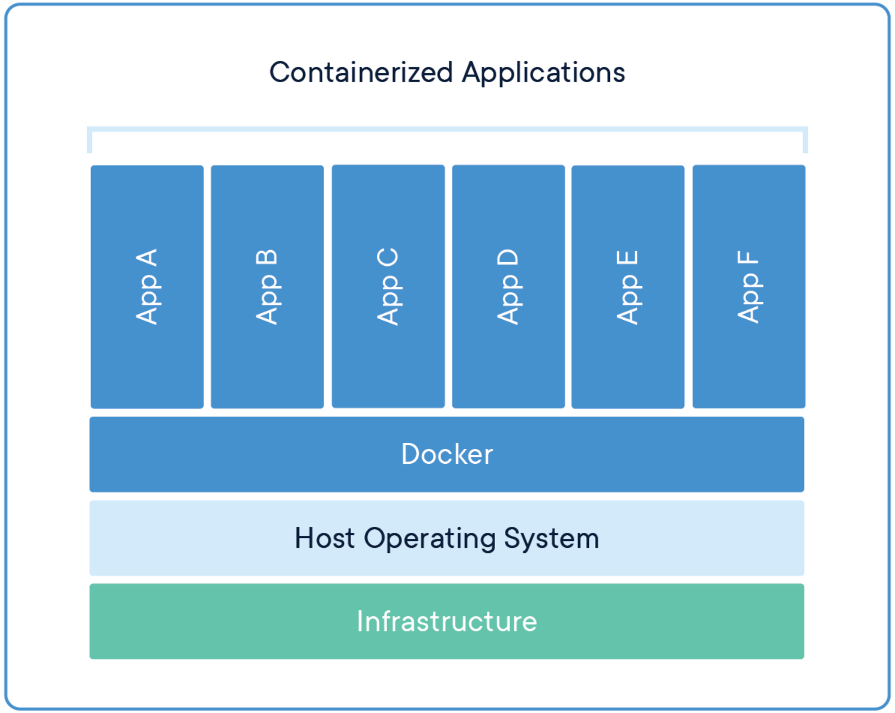
- Form of virtualization at the app packaging level (like virtual machines at the server level)
- Isolated from one another at the OS process layer (vs VM’s which are isolated at the hardware abstraction layer)
- Images represent the packaging up of an application and its dependencies as a complete, deployable unit of execution (code, runtime and configuration)

So, What Are Containers?

- A platform (e.g., Docker) running on a system can be used to dynamically create containers (executable instances of the app) from the defined image
- Typically, much, much smaller than a VM which makes them lightweight, quickly deployable and quick to “boot up”
- An orchestration engine (e.g., Kubernetes) might be used to coordinate multiple instances of the same container (or a “pod” of containers) to enable the servicing of more concurrent requests (scalability)

8

So, What Are Containers?



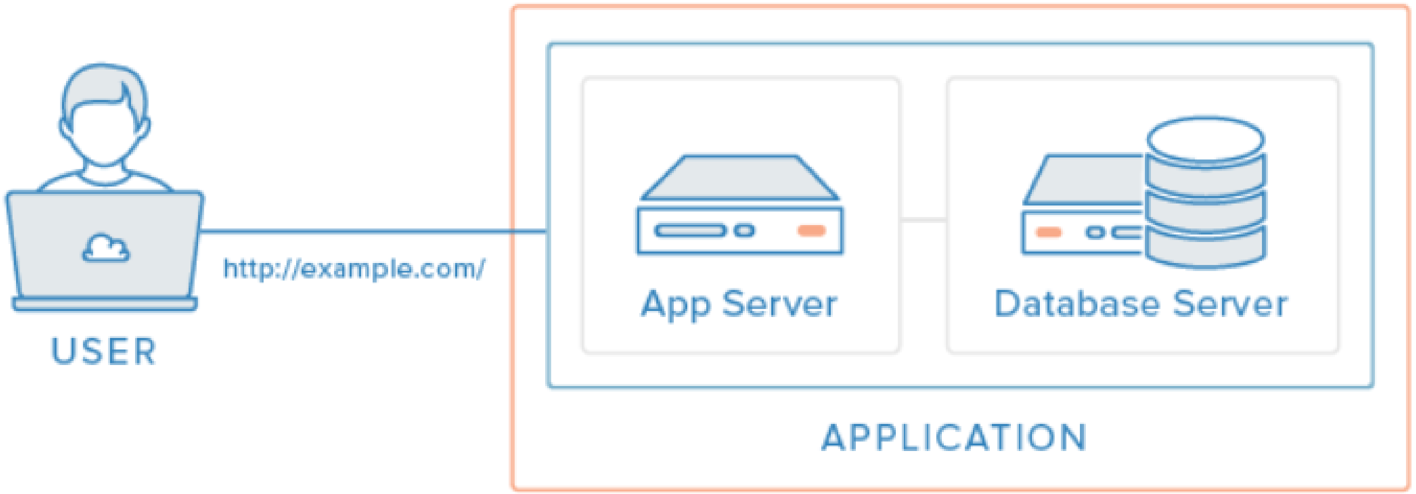
So, Why Containers?

- Abstracts away the code implementation so you can deploy in a platform-agnostic manner, writing in the language of your choice
- Aligns strongly with the principles and practices of DevOps
- Helps leverage the power of the cloud
- Speeds up important non-coding activities (infrastructure spin-up, testing, CI/CD tasks, DevSecOps, code quality checks, etc.
- Helps breed consistency vs. “snowflake”

10

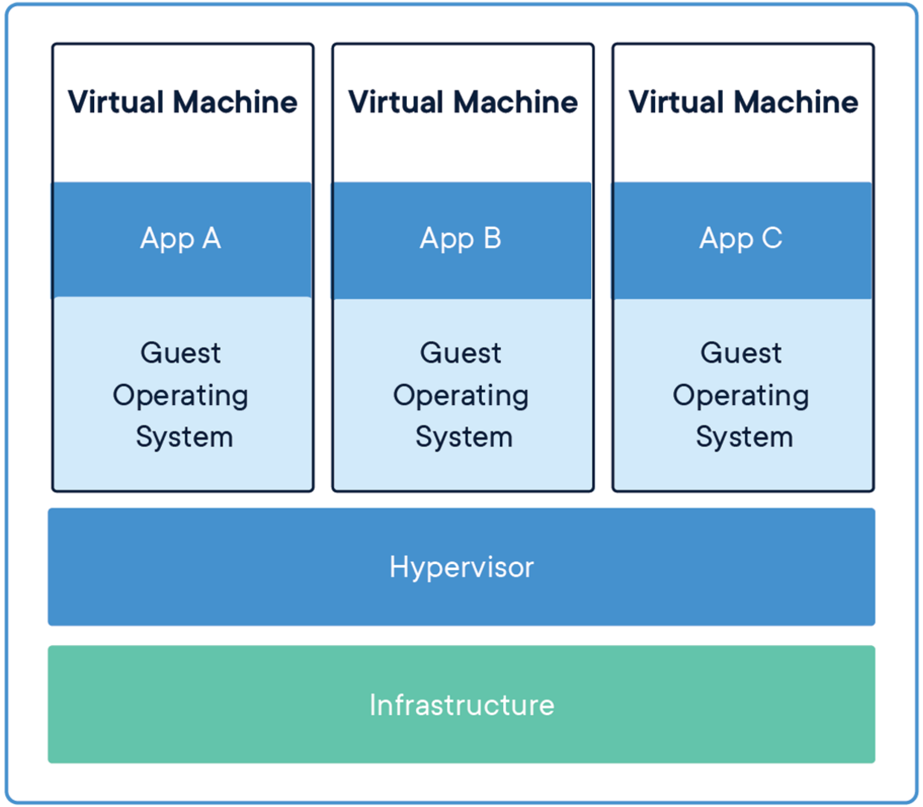
Docker as a Containerization Platform

Evolution of Containers – Client/Server



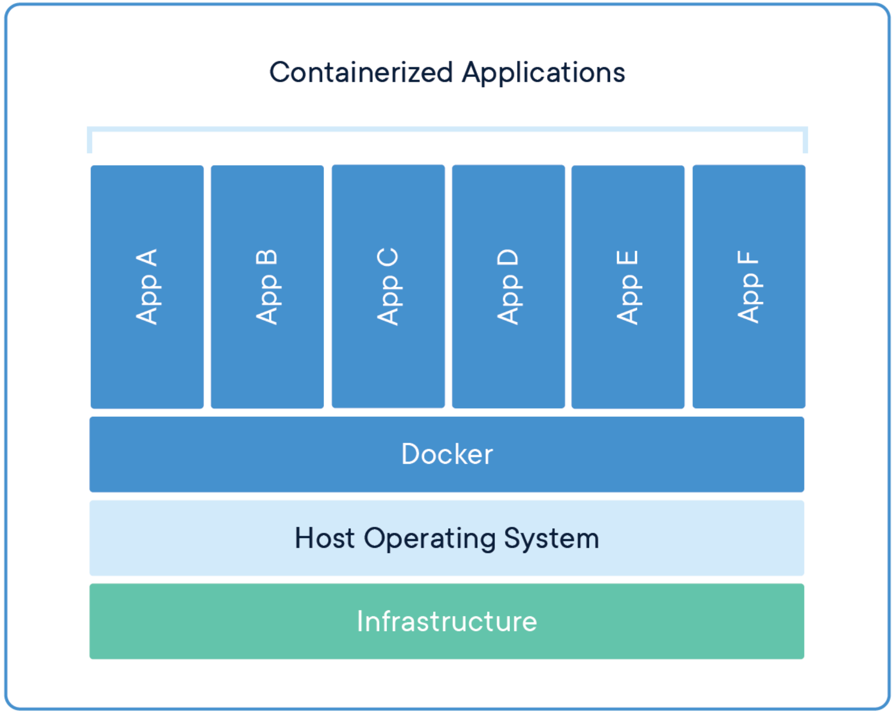
12

Evolution of Containers – Virtual Machines



13

Evolution of Containers – Containers



14

What is Docker?

- Open-source containerization technology
- Enables deployment of self-contained & isolated application instances
- One of the foundational technologies for supporting microservices

15

What is Docker?

- Built around the concept of images & containers
- Also, supports composition of a set of containers to be deployed together
- For example, application code + network components + database components

16

What is Docker?

- Utilizes principles of “immutable infrastructure”
- Complete application environments torn down and recreated as needed
- Helps to minimize infrastructure “drift” and environment inconsistencies

17

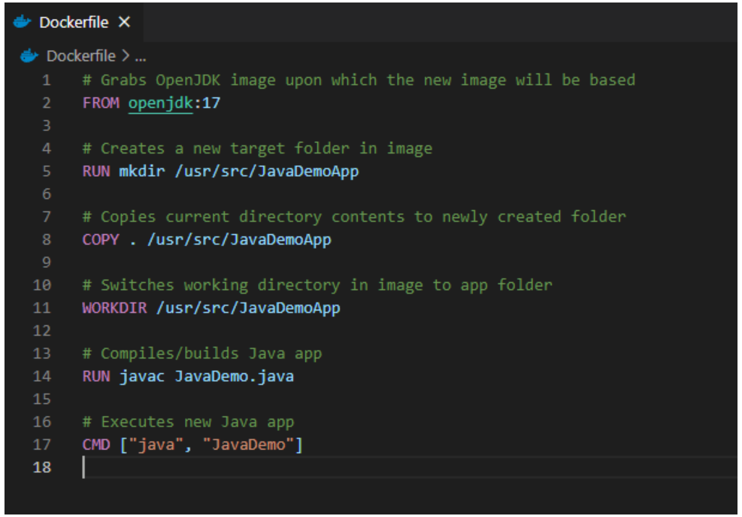
The Dockerfile

- Tells Docker what to do in creating an image for your application
- The commands are all things you could do from the CLI
- Used by the docker “build” command
- Docker build uses this file and a “context” – a set of files at a specified location – to make your image

18

Dockerfile example

The following creates an image for building/running Java app in container
See https://github.com/KernelGamut32/dockerlab-repo-sample for sample



```dockerfile
# Grabs OpenJDK image upon which the new image will be based
FROM openjdk:17

# Creates a new target folder in image
RUN mkdir /usr/src/JavaDemoApp

# Copies current directory contents to newly created folder
COPY . /usr/src/JavaDemoApp

# Switches working directory in image to app folder
WORKDIR /usr/src/JavaDemoApp

# Compiles/builds Java app
RUN javac JavaDemo.java

# Executes new Java app
CMD ["java", "JavaDemo"]
```

19

Docker Images

- Represent templates defining an application environment
- New instances of the application can be created from the image
- These instances are called containers

20

Docker Images

- Images are defined via a Dockerfile definition
- Support layers for building up the environment in stages
- Fully defines the application, including all components required to support

21

Docker Images

- Those components can include:
  - Runtime
  - Development framework
  - Source code
  - Executable instructions for container startup

22

Docker Images

- Start with a base that gives your app a place to live
    - Needed OS/runtimes/dB server applications, etc.
- Examples:
    - nginx
    - Node
    - MySQL
    - Apache HTTP Server
    - IIS with .NET Runtimes

23

Docker Hub

- Centralized registry for image storage & sharing
- Can signup for an account – user accounts offer both free and pro versions
- Also, supports organizations for grouping of multiple team members

24

Docker Hub

- Accessible at https://hub.docker.com
- Search feature enables search for image by technology or keyword
- Image detail displays available tags and image variants

25

Docker Hub

- May also include examples of usage
- Pro account supports image scanning for security vulnerabilities
- Can be useful for image reuse and image sharing across a dev team

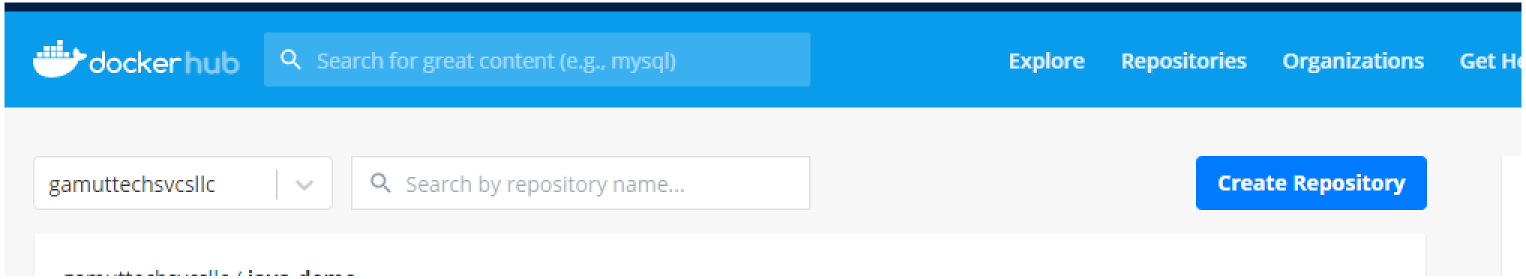
26

Docker Hub

- There are other registry types, including private registries

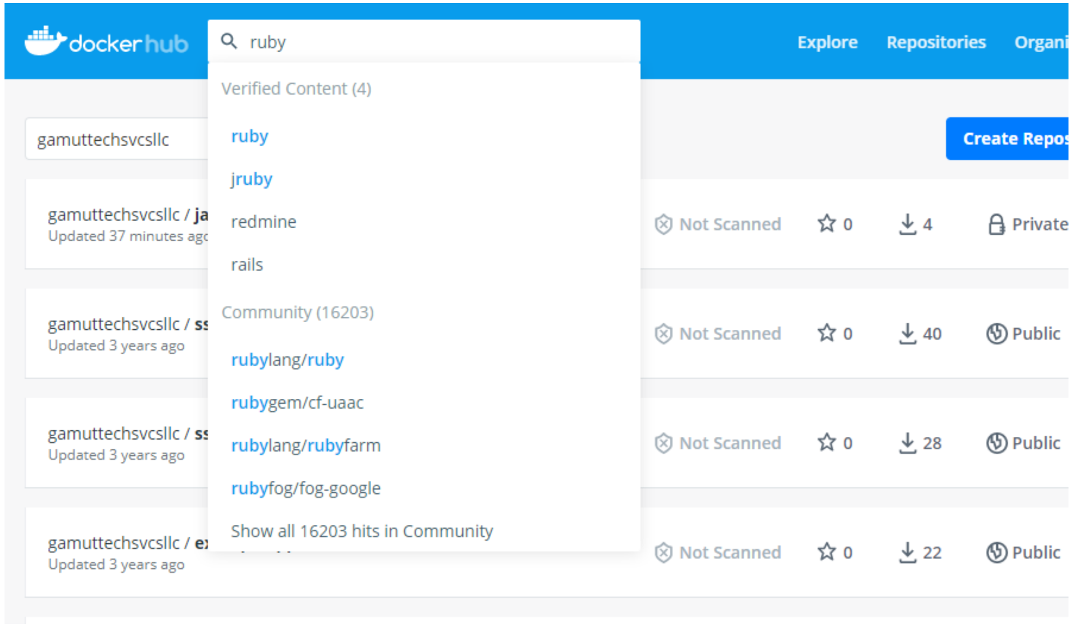
27

Docker Hub



28

Docker Hub



gamuttechsvcsllc

Verified Content (4)

ruby
jruby
redmine
rails

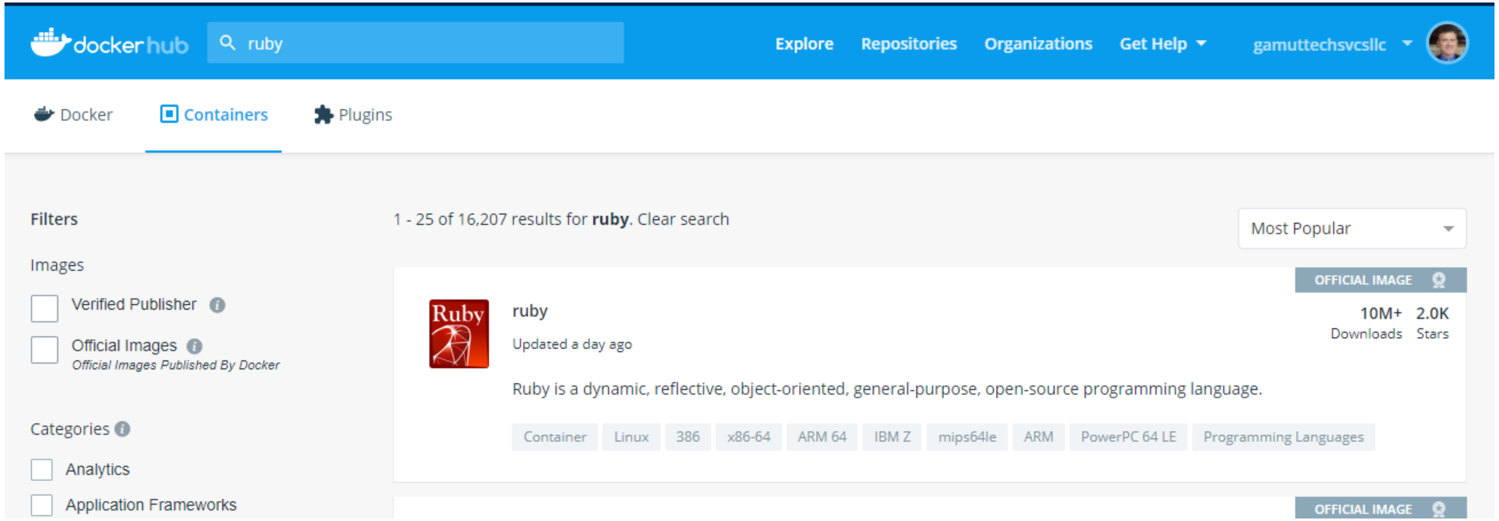
Community (16203)

rubylang/ruby
rubygem/cf-uaac
rubylang/rubyfarm
rubyfog/fog-google
Show all 16203 hits in Community

gamuttechsvcsllc / ja Updated 37 minutes ago	Not Scanned	0	4	Private
gamuttechsvcsllc / ss Updated 3 years ago	Not Scanned	0	40	Public
gamuttechsvcsllc / ss Updated 3 years ago	Not Scanned	0	28	Public
gamuttechsvcsllc / ex Updated 3 years ago	Not Scanned	0	22	Public

29

Docker Hub



Filters

Images

Verified Publisher

Official Images
Official Images Published By Docker

Categories

Analytics

Application Frameworks

1 - 25 of 16,207 results for **ruby**. Clear search

Most Popular

OFFICIAL IMAGE

ruby

Updated a day ago

10M+ Downloads 2.0K Stars

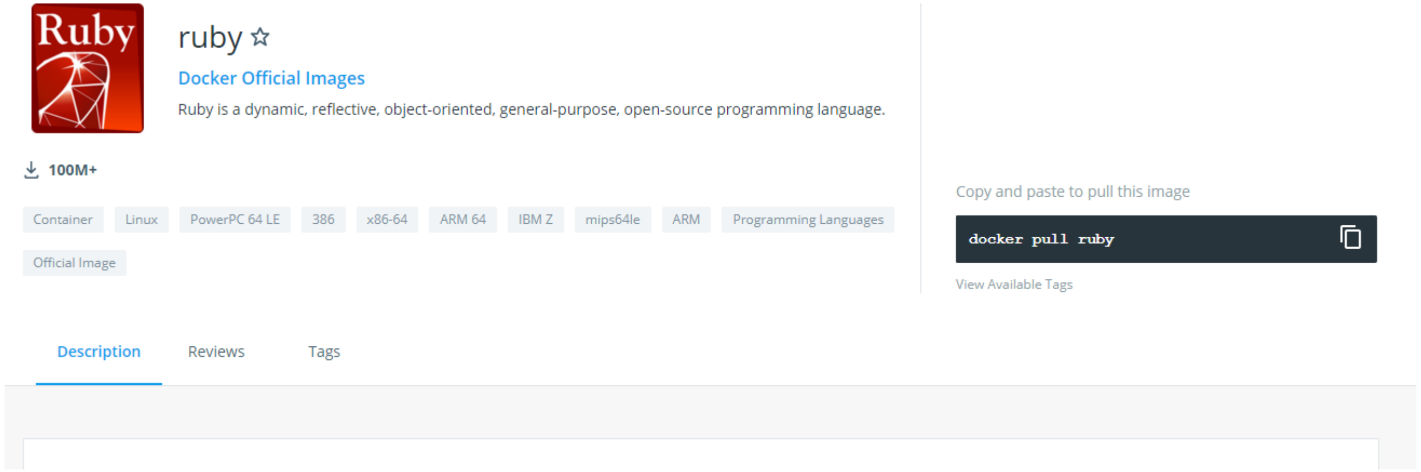
Ruby is a dynamic, reflective, object-oriented, general-purpose, open-source programming language.

Container | Linux | 386 | x86-64 | ARM 64 | IBM Z | mips64le | ARM | PowerPC 64 LE | Programming Languages

OFFICIAL IMAGE

30

Docker Hub



ruby ☆

Docker Official Images

Ruby is a dynamic, reflective, object-oriented, general-purpose, open-source programming language.

100M+

Container | Linux | PowerPC 64 LE | 386 | x86-64 | ARM 64 | IBM Z | mips64le | ARM | Programming Languages | Official Image

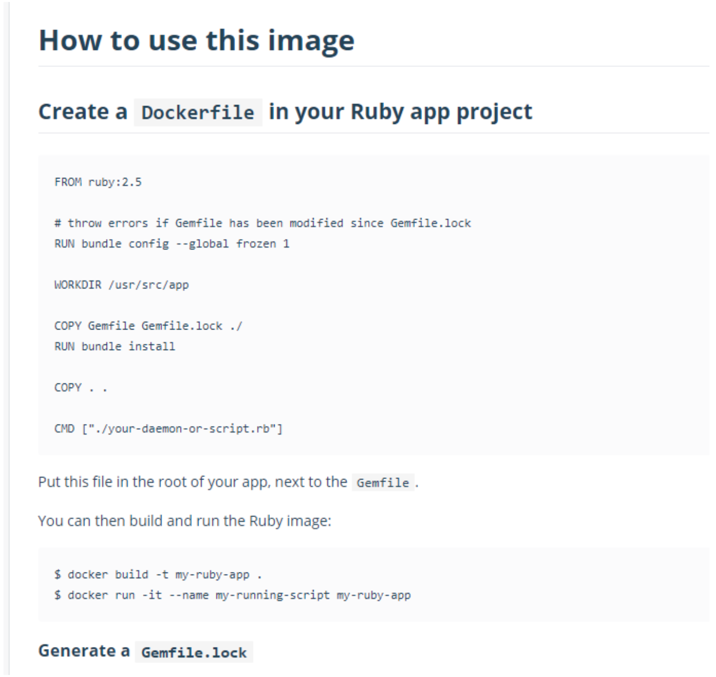
Copy and paste to pull this image

```
docker pull ruby
```

View Available Tags

31

Docker Hub



How to use this image

Create a `Dockerfile` in your Ruby app project

```
FROM ruby:2.5

# throw errors if Gemfile has been modified since Gemfile.lock
RUN bundle config --global frozen 1

WORKDIR /usr/src/app

COPY Gemfile Gemfile.lock ./
RUN bundle install

COPY . .

CMD ["./your-daemon-or-script.rb"]
```

Put this file in the root of your app, next to the `Gemfile`.

You can then build and run the Ruby image:

```
$ docker build -t my-ruby-app .
$ docker run -it --name my-running-script my-ruby-app
```

Generate a `Gemfile.lock`

32

Building The Image

- To build the image from Dockerfile use *docker build*
- *docker build -t <tag name> <path to Dockerfile>*
- For example, *docker build -t java-demo .*
- Builds image from Dockerfile in current folder (.) with tag name “java-demo”

33

Building The Image

- For tag name, can include optional detail:
  - Docker ID in Docker Hub for eventual push to image registry
  - Version identifier for tag – defaults to “latest” if excluded
- For example, *docker build -t <docker ID>/<tag name>:<version> .*

34

Pushing/Pulling Image

- Use *docker push <docker ID>/<tag name>:<version>* to push to registry
- Use *docker pull <docker ID>/<tag name>:<version>* to pull from registry
- May prompt for credentials (e.g., Docker Hub login)

35

Pushing/Pulling Image

- To pull from public registry, use docker pull <tag name>:<version>
- For example, *docker pull openjdk:17*
- A .dockerignore file can be used to omit files/folders on push

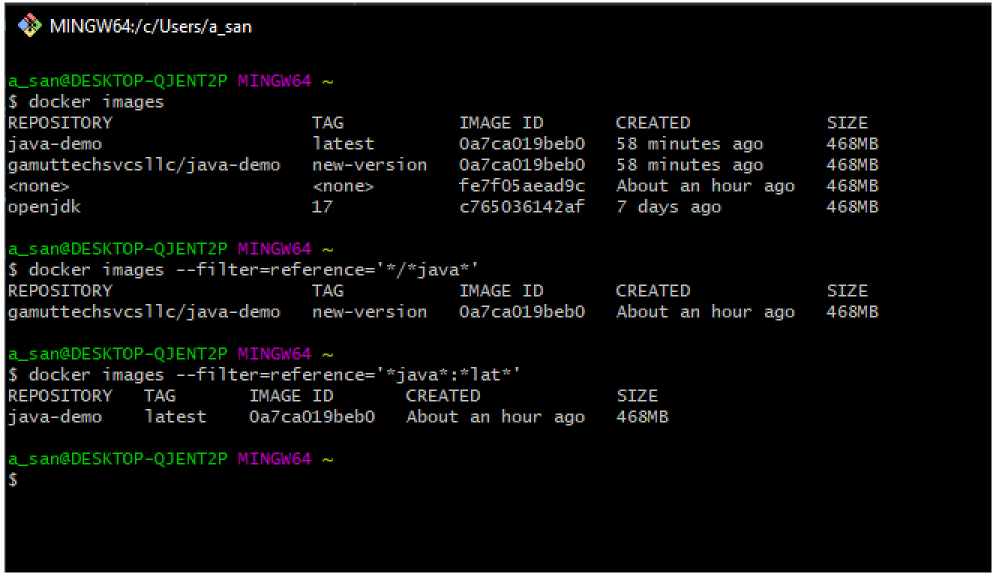
36

Managing Images

- Use *docker images* to list available local images
- --*filter* argument enables wildcard search
- *docker images* --*filter=reference=‘<wildcard for tag name>:<wildcard for version>’*

37

Managing Images


MINGW64:/c/Users/a_san

```
a_san@DESKTOP-QJENT2P MINGW64 ~
$ docker images
REPOSITORY                   TAG           IMAGE ID       CREATED              SIZE
java-demo                    latest        0a7ca019beb0   58 minutes ago       468MB
gamuttechsvcsllc/java-demo   new-version   0a7ca019beb0   58 minutes ago       468MB
<none>                       <none>        fe7f05aead9c   About an hour ago    468MB
openjdk                      17            c765036142af   7 days ago           468MB

a_san@DESKTOP-QJENT2P MINGW64 ~
$ docker images --filter=reference='*/*java*'
REPOSITORY                   TAG           IMAGE ID       CREATED              SIZE
gamuttechsvcsllc/java-demo   new-version   0a7ca019beb0   About an hour ago    468MB

a_san@DESKTOP-QJENT2P MINGW64 ~
$ docker images --filter=reference='*java*:*lat*'
REPOSITORY   TAG       IMAGE ID       CREATED              SIZE
java-demo    latest    0a7ca019beb0   About an hour ago    468MB

a_san@DESKTOP-QJENT2P MINGW64 ~
$
```

38

Managing Images

- To remove an image, use *docker rmi*
- *-f* argument used to remove images even when used by containers
- Can use *docker rmi -f <image ID>* to remove specific image

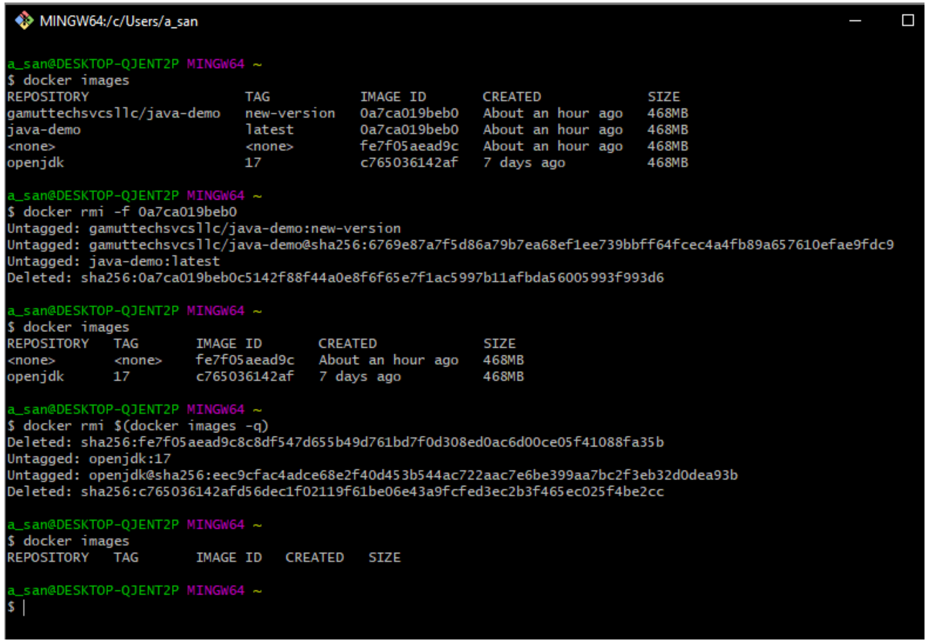
39

Managing Images

- Can use *docker rmi -f $(docker images -q)* to remove all (CAUTION)
- *docker images -q* lists images in quiet mode (returns image ID's only)

40

Managing Images


MINGW64:/c/Users/a_san

```
a_san@DESKTOP-QJENT2P MINGW64 ~
$ docker images
REPOSITORY                   TAG           IMAGE ID       CREATED              SIZE
gamuttechsvcsllc/java-demo   new-version   0a7ca019beb0   About an hour ago    468MB
java-demo                    latest        0a7ca019beb0   About an hour ago    468MB
<none>                       <none>        fe7f05aead9c   About an hour ago    468MB
openjdk                      17            c765036142af   7 days ago           468MB

a_san@DESKTOP-QJENT2P MINGW64 ~
$ docker rmi -f 0a7ca019beb0
Untagged: gamuttechsvcsllc/java-demo:new-version
Untagged: gamuttechsvcsllc/java-demo@sha256:6769e87a7f5d86a79b7ea68ef1ee739bbff64fcec4a4fb89a657610efae9fdc9
Untagged: java-demo:latest
Deleted: sha256:0a7ca019beb0c5142f88f44a0e8f6f65e7f1ac5997b11afbda56005993f993d6

a_san@DESKTOP-QJENT2P MINGW64 ~
$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED             SIZE
<none>       <none>    fe7f05aead9c   About an hour ago   468MB
openjdk      17        c765036142af   7 days ago          468MB

a_san@DESKTOP-QJENT2P MINGW64 ~
$ docker rmi $(docker images -q)
Deleted: sha256:fe7f05aead9c8c8df547d655b49d761bd7f0d308ed0ac6d00ce05f41088fa35b
Untagged: openjdk:17
Untagged: openjdk@sha256:eec9cfac4adce68e2f40d453b544ac722aac7e6be399aa7bc2f3eb32d0dea93b
Deleted: sha256:c765036142afd56dec1f02119f61be06e43a9fcfed3ec2b3f465ec025f4be2cc

a_san@DESKTOP-QJENT2P MINGW64 ~
$ docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE

a_san@DESKTOP-QJENT2P MINGW64 ~
$ |
```

41

Layers in a Docker Image

- Each instruction in a dockerfile makes a “layer”
- It’s actually a diff from the earlier layer
- Allows Docker to skip redundant info and use cached artifacts
- In this way, images themselves are actually diffs – they show what changed from the earlier stage (e.g., our app’s image is actually a diff from the base image you chose)

42

Best Practices for Creating Docker Images

- Single app per container
- Don’t include unnecessary tools in your image (dev tools; network tools like netcat, etc.)
- Build as small an image as possible
    - Choose a small base image
    - Optimize your app for image size
- Use a consistent “tag” strategy
    - Document it
    - Use it for version info, testing strategy info, etc.
- Be smart about using public base images

43

Troubleshooting Docker Images

- Most common area is the Dockerfile
- Error on docker build has good messaging and an error code

44

Docker Containers

- Represent “runnable” instances of a docker image
- Application instance created from image in container can be used as:
  - Isolated executable
  - Request servicer (e.g., web listener)

45

Docker Containers

- Includes all dependencies and runtime defined by the image
- Isolated from other containers at the OS process layer
- Mechanisms exist to share resources across containers (e.g., files or DBs)
- However, isolation is what makes them powerful – avoid unnecessary coupling

46

Docker Containers

- Typically, containers are much smaller which makes them lightweight
- Quickly deployable and quick to “boot up”
- Isolation allows technologies like k8s to spin up multiple as needed
- “Load balancers” route to any of multiple instances using single point of connection

47

Creating Containers

- To create a new container, you can use *docker create*
- Multiple options are provided for configuring container (see *docker create --help*)
- For example, *docker create --name <container name> <image tag>*
- *<image tag>* defines image (template) from which to create container

48

Creating Containers

- To list available containers, use *docker ps*
- *docker ps -a* lists all containers (even those not currently started)
- Containers can be stopped and started

49

Creating Containers

- Use *docker start <name>* or *docker start <container ID>* to start
- Use *docker stop <name>* or *docker stop <container ID>* to stop
- Use *docker run* to create and start container in single step
- *docker run* is the more common command

50

Configuring Containers

- Command-line options for configuring containers using *docker run* include:
  - *--name <container name>* – give container user-defined name
  - *-p <host port>:<container port>* – map host port to container port for access
  - *-it* – indicates interactive on command-line (e.g., for gathering command-line input)
  - *--rm* – container automatically deleted when it exits or stops
  - *-d* – container runs in detached mode (e.g., for continually running web listeners)
- See *docker run --help* for additional info

51

Container Status

- Use *docker logs <container name>* to see log output from container
- Use *docker logs -f <container name>* for ongoing monitor of log output
- Helpful for troubleshooting issues with container creation, startup, or operation

52

Container Commands

- *docker exec* can be used to execute a command in a running container
- For example, *docker exec <container name> ls -a* to see container file contents
- *docker exec -it <container name> /bin/bash* for interactive command-line session in container (for Linux-based images that support bash)

53

Managing Containers

- *docker rm <container name>* or *docker rm <container ID>* removes container
- *-f* argument used to remove a running container
- Can use *docker rm -f $(docker ps -aq)* to remove all (CAUTION)
- *docker ps -aq* lists all containers in quiet mode (returns container ID’s only)

54

Data Storage in Docker

- Running containers generate data
- They may need access to "persistent" data
- We can use the host machine via a "bind mount" – mount a local file or folder into a container
  - Problems: Not easily managed by docker CLI
  - Rely on the host machine having a specific directory structure
- Better is to use a docker construct called a "volume"
  - New directory created in the host machine's docker storage directory
  - Easier to back up
  - Work the same on both Linux and Windows containers
  - Can be safely shared between containers
  - Content can be pre-populated by the container

55

Common Docker Issues – How to Identify and Fix

- Dependency issues with base image:
    - RUN apt-get clean && apt-get update (clears cache in event base image has been updated in the registry)
- You may need to do this outside the container as well
- Container naming collisions:
    - If you try to use a container name that exists, you'll throw an error – EVEN if the container isn't being used. Remove it to use the name again.

56

Application Bootstrapping with Docker and k8s

- Kubernetes provides a hosting environment for containerized applications
- Once you have a Docker image, you can work entirely within Kubernetes to deploy your app

57

Open Container Initiative (OCI)

- The OCI is an open governance structure for the express purpose of creating open industry standards around container formats and runtimes
- Docker started it
- They have two specs: runtime-spec and image-spec
- This deals with containers in the abstract

58

Competing Container Runtimes

- rkt from CoreOS
- Mesos from Apache
- LXC Linux containers

59

Azure Container Registry (ACR)

Azure Container Registry (ACR)

https://learn.microsoft.com/en-us/azure/container-registry/container-registry-intro

ACR Tasks

https://learn.microsoft.com/en-us/azure/container-registry/container-registry-tasks-overview

https://learn.microsoft.com/en-us/cli/azure/acr?view=azure-cli-latest#az-acr-create

https://learn.microsoft.com/en-us/cli/azure/acr?view=azure-cli-latest#az-acr-build

https://learn.microsoft.com/en-us/cli/azure/acr?view=azure-cli-latest#az-acr-run

LAB:

ACR Tasks

Execute the “Hands-On” lab available at
https://github.com/KernelGamut32/azure_docker_microservices-public/tree/main/week02/labs/lab01

Azure Container Instances (ACI)

Azure Container Instances (ACI)

https://learn.microsoft.com/en-us/azure/container-registry/container-registry-intro

LAB:

ACI

Execute the “Hands-On” lab available at https://github.com/KernelGamut32/azure_docker_microservices-public/tree/main/week02/labs/lab02

LAB:

Using Azure Container Instances

Execute the “Hands-On” lab available at https://github.com/KernelGamut32/azure_docker_microservices-public/tree/main/week02/labs/lab03

LAB:

Containerized Python with MongoDB

Execute the “Hands-On” lab available at https://github.com/KernelGamut32/azure_docker_microservices-public/tree/main/week02/labs/lab04

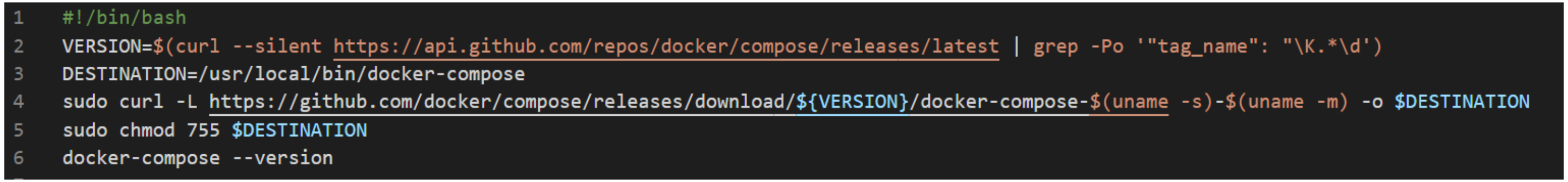
Docker Compose

Docker Compose as a Dev Tool

- Used to run multi-container applications
- Declaratively configures your app’s services as a unit
- All services can be started with one command

Docker Compose as a Dev Tool

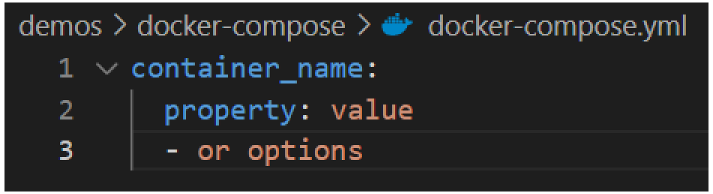
- May require separate install
- To check for presence, run `*docker-compose --version*`
- To install:



```bash
#!/bin/bash
VERSION=$(curl --silent https://api.github.com/repos/docker/compose/releases/latest | grep -Po '"tag_name": "\K.*\d')
DESTINATION=/usr/local/bin/docker-compose
sudo curl -L https://github.com/docker/compose/releases/download/${VERSION}/docker-compose-$(uname -s)-$(uname -m) -o $DESTINATION
sudo chmod 755 $DESTINATION
docker-compose --version
```

Docker Compose as a Dev Tool

- docker-compose.yml used to define containers and relationships
- Uses YAML (Yet Another Markup Language)
- General format:



demos > docker-compose > docker-compose.yml

```
container_name:
  property: value
  - or options
```

Docker Compose as a Dev Tool

- Supports all properties available with `*docker run*`
- Uses a `*links*` property to link two containers together
- In it, specify required connections to existing container definition

Docker Compose as a Dev Tool

- Define secondary container(s) using same format
- Linked by identifier
- Multiple containers can be defined together in single YAML file

Docker Compose as a Dev Tool

- `*docker-compose up*` uses YAML file to launch all containers with one command
- Use `*docker-compose up <name>*` to bring up single container
- `*-d*` argument runs in background (similar to use with `*docker run*`)

Docker Compose as a Dev Tool

- `*docker-compose ps*` displays details for all launched containers
- `*docker-compose logs*` display all logs for multi-container “unit”
- `*docker-compose scale web=#*` scales the number of web containers (use 1 to scale back down)

Docker Compose as a Dev Tool

- `*docker-compose stop*` to stop all containers
- `*docker-compose down*` or `*docker-compose rm*` to remove all containers

- See https://docs.docker.com/compose/compose-file/

Multi-stage Container Builds

- By setting a dependency in your Docker Compose file, you can make your multi-container app spin up in a specific order.

Demo: Using Docker Compose

DEMO:

Docker Compose

Execute the "Hands-On" lab available at https://github.com/KernelGamut32/azure_docker_microservices-public/tree/main/week02/demos/docker-compose

LAB:

Multi-Container Group Using a YAML File

Execute the “Hands-On” lab available at https://github.com/KernelGamut32/azure_docker_microservices-public/tree/main/week02/labs/lab05

LAB:

Using Azure Automated ML

Execute the “Hands-On” lab available at https://github.com/KernelGamut32/azure_docker_microservices-public/tree/main/week02/labs/lab06

LAB:

Service Principal Authentication for ACR

Execute the “Hands-On” lab available at
https://github.com/KernelGamut32/azure_docker_microservices-public/tree/main/week02/labs/lab07

Thank you!

If you have additional questions, please reach out to me at: asanders@gamuttechnologysvcs.com

PLURALSIGHT