Agenda

- Containers and VMs
- Docker
  - Docker Images
  - Docker Containers
  - Dockerfiles
- Managing Data in Docker
  - Volumes
  - Bind Mounts
  - tmpfs
- Docker Registry
- Docker in the Cloud
- Demo!
“Now that you have an overview of the system, we’re ready for a little more detail”
Google Trends for “Docker” and “Virtual Machine”
Containers vs Virtual Machines
Docker

- Docker is a platform for developers and sysadmins to **develop, deploy, and run applications with containers**.
- Initial release was in March, 2013
- Containerization is
  - Flexible
  - Lightweight
  - Portable
  - Scalable
  - Stackable
## Docker CE vs Docker Enterprise

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Container engine and built in orchestration, networking, security</td>
<td>☑</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Certified infrastructure, plugins and ISV containers</td>
<td></td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Image management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Container app management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image security scanning</td>
<td></td>
<td></td>
<td></td>
<td>☑</td>
</tr>
</tbody>
</table>
Docker Image

- Package that includes everything needed to run an application.
- Organized in *layers*.
- Used to build a containers.
- *docker images* lists all images.
- Docker Hub is a repository of images.
Docker Container

- Wraps an application’s software into an invisible box with everything the application needs to run.
- Built on top of images (read-only) and then adds a read-write file system.
- You can list all running containers with command `docker ps -a`
Dockerfile

- **Instructions to build a Docker image.**
- Defines what goes on in the environment inside your container and access to resources.
- You can create an image from a Dockerfile with `docker build` command.
### Dockerfile Commands*

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>DESCRIPTION</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>Copies the files from the source on the host into the container’s own filesystem at the set destination.</td>
<td>ADD /home/my_app /my_app</td>
</tr>
<tr>
<td>CMD</td>
<td>Executes a specific command when a container is instantiated.</td>
<td>CMD &quot;echo&quot; &quot;Hello docker!&quot;</td>
</tr>
<tr>
<td>ENV</td>
<td>Sets the environment variable &lt;key&gt; to the value &lt;value&gt;</td>
<td>ENV level=&quot;Leeeeroy Jeeeeeeeeenkins&quot;</td>
</tr>
<tr>
<td>EXPOSE</td>
<td>Enables networking between the running process inside the container and the outside world.</td>
<td>EXPOSE 8080</td>
</tr>
<tr>
<td>FROM</td>
<td>Defines the base image.</td>
<td>FROM ubuntu</td>
</tr>
<tr>
<td>RUN</td>
<td>Runs a given command during the build process.</td>
<td>RUN apt-get install -y riak</td>
</tr>
<tr>
<td>VOLUME</td>
<td>Enable access from your container to a directory on the host machine.</td>
<td>VOLUME [&quot;/my_files&quot;]</td>
</tr>
</tbody>
</table>

* not exhaustive list
Dockerfile Example

# Our base image
FROM alpine:3.5

# Install python and pip
RUN apk add --update py2-pip

# Install Python modules needed by the Python app
COPY requirements.txt /usr/src/app/
RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt

# Copy files required for the app to run
COPY app.py /usr/src/app/
COPY templates/index.html /usr/src/app/templates/

# Tell the port number the container should expose
EXPOSE 5000

# Run the application
CMD ["python", "/usr/src/app/app.py"]
Managing Data in Docker

- You can store data in the *writeable layer* of the container, but:
  - It’s **not persistent**
  - Tightly coupled to the host
  - Reduced performance
- Three ways to mount data into a container
  - Volumes
  - Bind mounts
  - tmpfs volumes.
- Volumes are almost always the right choice
Managing Data in Docker (cont.)

- **Volumes** are stored in a part of the host filesystem which is *managed by Docker*. Non-Docker processes should not modify this part of the filesystem.
- **Bind mounts** may be stored *anywhere* on the host system. Non-Docker processes on the Docker host or a Docker container can modify them at any time.
- **tmpfs mounts** are stored in the host system’s memory only, and are never written to the host system’s filesystem.
Volumes

- Created and managed by Docker
  - `docker volume create`
  - `docker volume prune`
  - During container creation
- Stored within a directory in host, then mounted in the container
- A given volume can be mounted into multiple containers simultaneously.
- Can be named or anonymous
- If in doubt, always choose volumes.
Bind Mounts

- File or directory on the host machine is mounted into a container.
  - Referenced by its full path
  - Created if it doesn’t exist yet
- Bind mounts are very performant, but they rely on the host machine’s filesystem having a specific directory structure available.
- Typically used for sharing configuration files or source code between the host and the container (in development environment).
tmpfs

- **Not persisted** on disk.
- Can be used to store non-persistent state or sensitive information.
  - Security reasons
  - Protect the performance of the container when your application needs to write a large volume of non-persistent state data.
Docker Registry

- It is an application that stores and lets you distribute Docker images.
- The Registry is open-source, under the permissive Apache license.
- Docker Hub
Docker in the Cloud

- Amazon Web Services
- Azure
- Heroku
- Google Cloud Platform
DEMO!
Thank you very much!