Ibm Uvt Proiect Colectiv Devops

IBM UVT Proiect Colectiv DevOps

This repository holds course materials for "Podman" course, part of `DevOps UVT Proiect Colectiv`.

Course Structure

- 1-introduction-to-containers-and-podman (teoretical section)
  - What are containers?
  - Containers vs Virtual Machines
  - Podman
  - Dockerfile (definition) to Image (build) to Containers (running)
  - Writing Containerfiles (Dockerfiles)
  - Container Registries
  - Further Reading Materials
- 2-hands-on-exercises (practical section)
  - Install Podman (Rokcylinux)
  - Create account on Dockerhub.com
  - Build an image
  - Tag an image
  - Pull an image from a public container registry
  - Push an image to a public container registry
- 3-container-orchestration (OPTIONAL: If we have time left)
  - What/Why?
  - Kubernetes and Openshift

Repository structure

...

1 Introduction To Containers And Podman

Containers

What are containers?

One of the bigger pain points that has traditionally existed between development and operations teams is how to make changes rapidly enough to support effective development but without risking the stability of the production environment and infrastructure.

A relatively new technology that helps alleviate some of this friction is the idea of software containers — isolated structures that can be developed and deployed relatively independently from the underlying operating system or hardware.

Similar to virtual machines, containers provide a way of sandboxing the code that runs in them, but unlike virtual machines, they generally have less overhead and less dependence on the operating system and hardware that support them.

This makes it easier for developers to develop an application in a container in their local environment and deploy that same container into production, minimizing risk and development overhead while also cutting down on the amount of deployment effort required of operations engineers.

Containers vs Virtual Machines

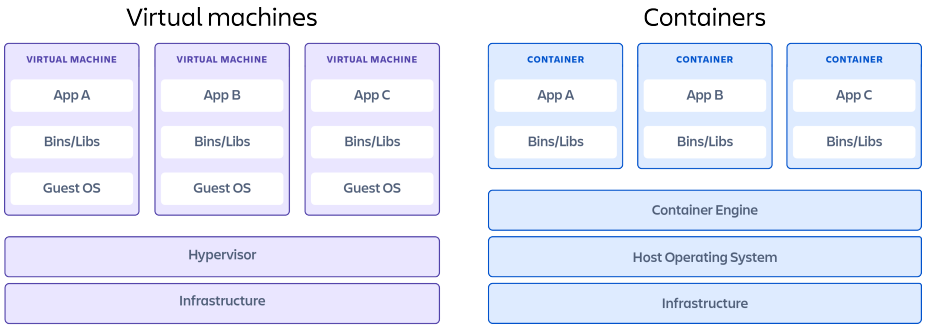

Figure 1: Containers vs VM illustration

In a **virtual machine (VM)**, we need to install an operating system with the appropriate device drivers; hence,the footprint or size of a virtual machine is huge. A normal VM with Tomcat and Java installed may take up to 10 GB of drive space: There's an overhead of memory management and device drivers. A VM has all the components a normal physical machine has in terms of operation.

In a VM, the hypervisor abstracts resources. Its package includes not only the application, but also the necessary binaries and libraries, and an entire guest operating system, for example, CentOS 6.7 and Windows 2003. Cloud service providers use a hypervisor to provide a standard runtime.

A **container** shares the operating system and device drivers of the host. Containers are created from images, and for a container with Tomcat installed, the size is less than 500 MB: Containers are small in size and hence effectively give faster and better performance. They abstract the operating system. A container runs as an isolated user space, with processes and filesystems in the user space on the host operating system itself, and it shares the kernel with other containers. Sharing and resource utilization are at their best in containers, and more resources are available due to less overhead. It works with very few required resources.

Podman

...

... basic Podman commands

...

Containerfile (definition) to Image (build) to Containers (running)

A Containerfile (Dockerfile) is the Docker image's source code. A Containerfile (Dockerfile) is a text file containing various instructions and configurations. The `FROM` command in a Containerfile (Dockerfile) identifies the base image from which you are constructing.

Writing Containerfiles (Dockerfiles)

A Containerfile (Dockerfile) is a text-based document that's used to create a container image. It provides instructions to the image builder on the commands to run, files to copy, startup command, and more.

Example:

```
FROM python:3.12

WORKDIR /usr/local/app

# Install the application dependencies
COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt

# Copy in the source code
COPY src ./src
EXPOSE 5000

# Setup an app user so the container doesn't run as the root user
RUN useradd app
USER app

CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8080"]
```

For a complete guide check Dockerfile Refference

Container Registries

An **image registry** is a centralized location for **storing and sharing** your container images. It can be either **public** or **private**. Docker Hub is a public registry that anyone can use and is the default registry.

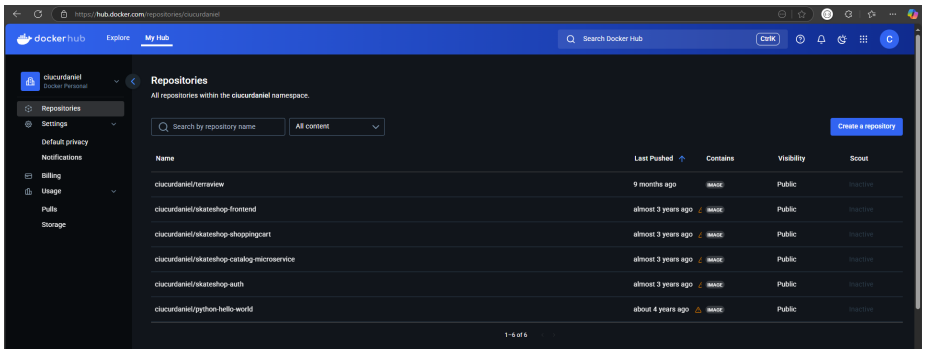

Figure 2: Screenshot from DockerHub

While Docker Hub is a popular option, there are many other available container registries available today, including Amazon Elastic Container Registry(ECR), Azure Container Registry (ACR), and Google Container Registry (GCR). You can even run your private registry on your local system or inside your organization. For example, Harbor, JFrog Artifactory, GitLab Container registry etc.

Further Reading Materials

- IBM Introduction to containerization
- Best practices for building containers
- Write your first Containerfile for Podman
- Base Images

2 Hands On Exercises

Hands on exercises

Install Podman (Rokcylinux)

Podman should come pre-installed in Rockylinux. If that is not the case, install it using:

```
dnf install podman
```

Verify installation:

```
podman version
```

Hint: Rockylinux - Podman Guide Rockylinux - Podman Guide - 2

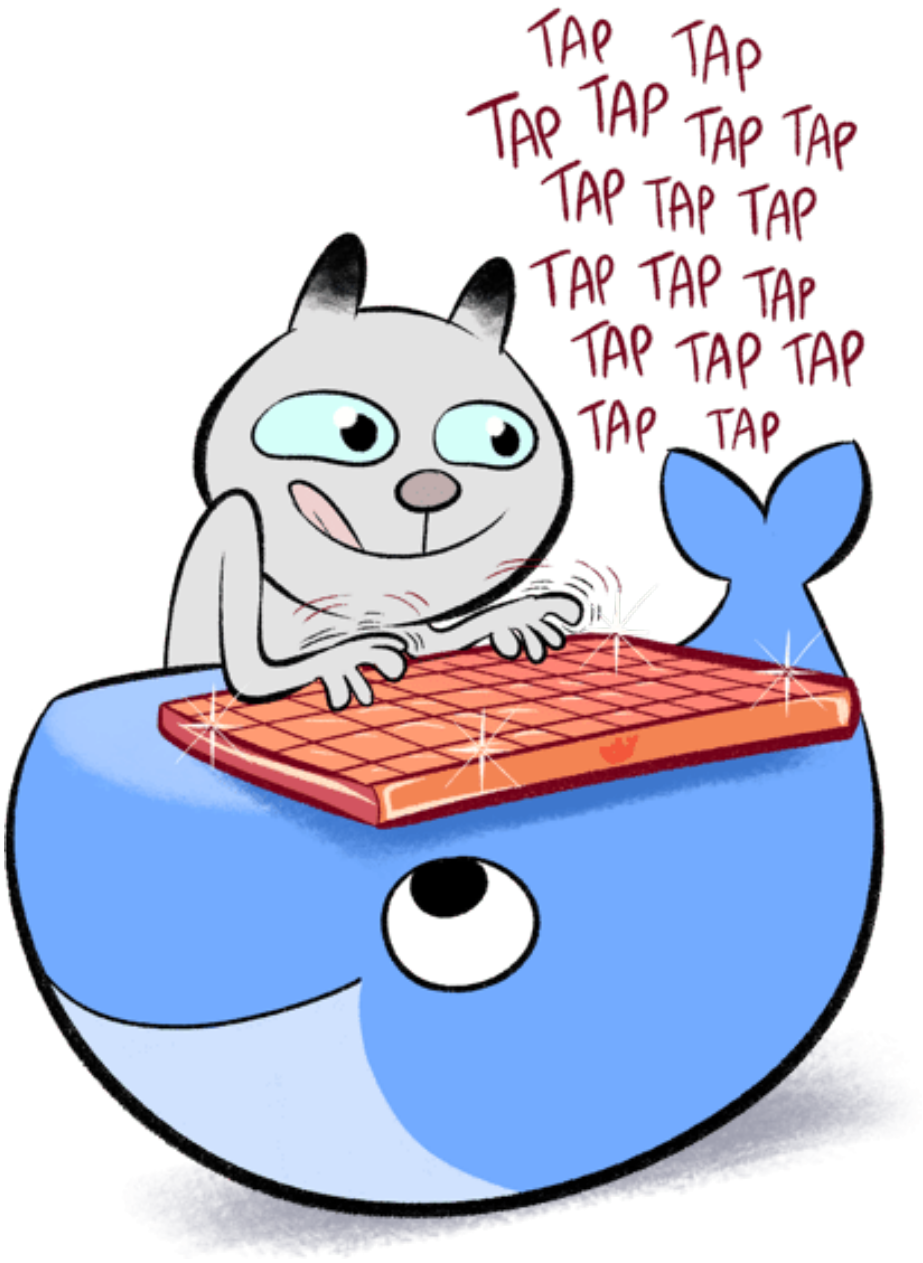

Figure 3: Cat with keyboard

Create an account on Dockerhub.com

Go to https://hub.docker.com and sign up for an account.

Remember the credentials! We will need them to push images to DockerHub

Build an image

...

Tag an image

...

Pull an image from a public container registry

...

Push an image to a public container registry

...

3 Container Orchestration

Container orchestration

What/Why?

Today an organization might have hundreds or thousands of containers. An amount that would be nearly impossible for teams to manage manually. This is where container orchestration comes in.

A container orchestration platform schedules and automates management like container deployment, networking, load balancing, scalability and availability.

- Provisioning
- Redundancy
- Health monitoring
- Resource allocation
- Scaling and load balancing
- Moving between physical hosts

Kubernetes and Openshift

...