Podman For The Absolute Beginners

By Thinknyx Technologies LLP

Yogesh Raheja

Puppet for the Absolute Beginners - Hands-on - DevOps

Infrastructure Automation with OpenTofu – Hands-On DevOps

SaltStack for the Absolute Beginners - Practical DevOps

AI Ecosystem for the Absolute Beginners - Hands-On

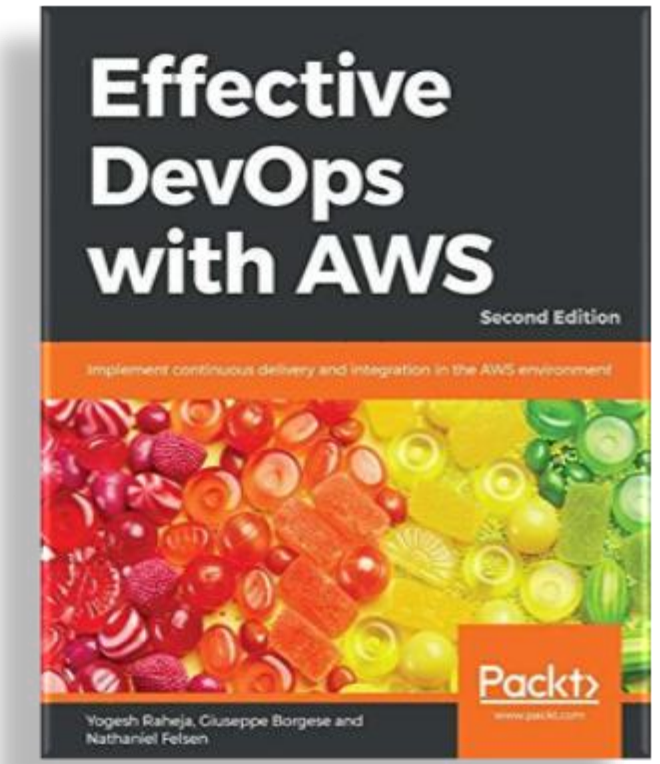
Mastering Prompt Engineering for GenAI - Practical Workshop

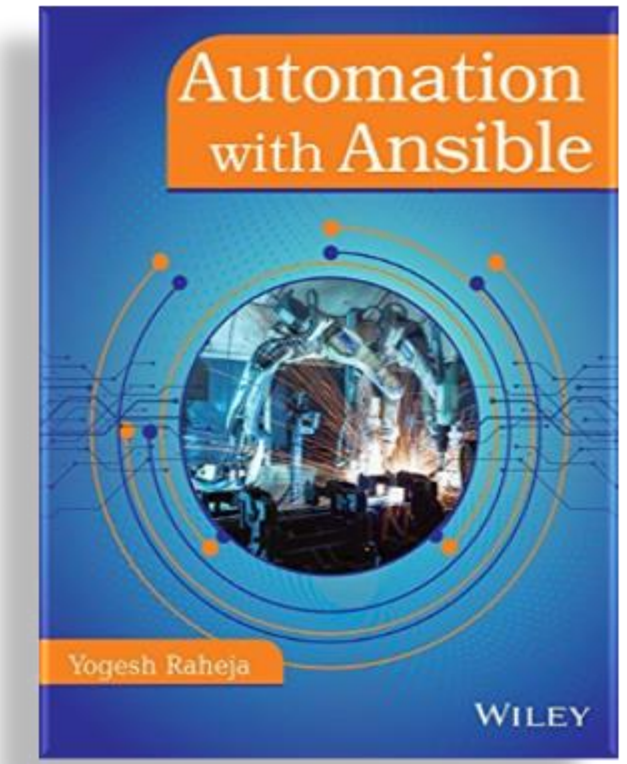
Generative AI Essentials - Practical Use Cases

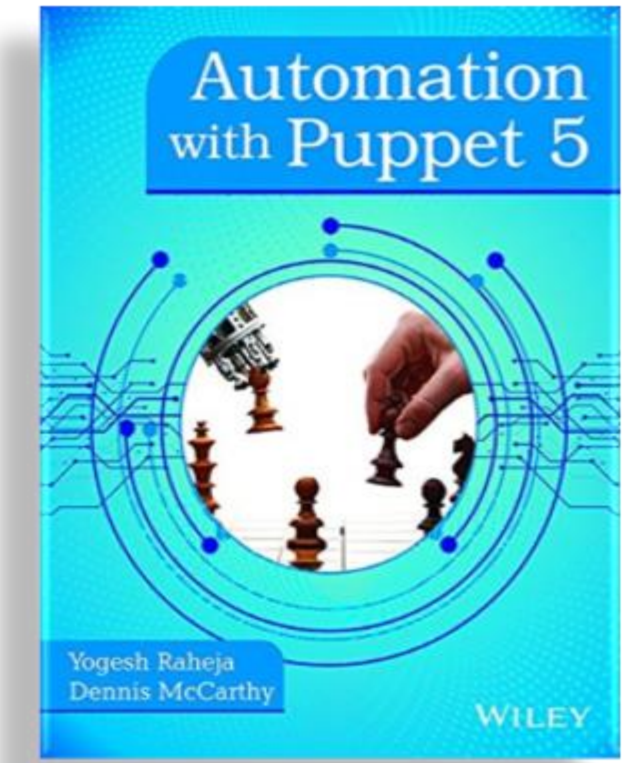
Mastering Docker Essentials - Hands-on DevOps

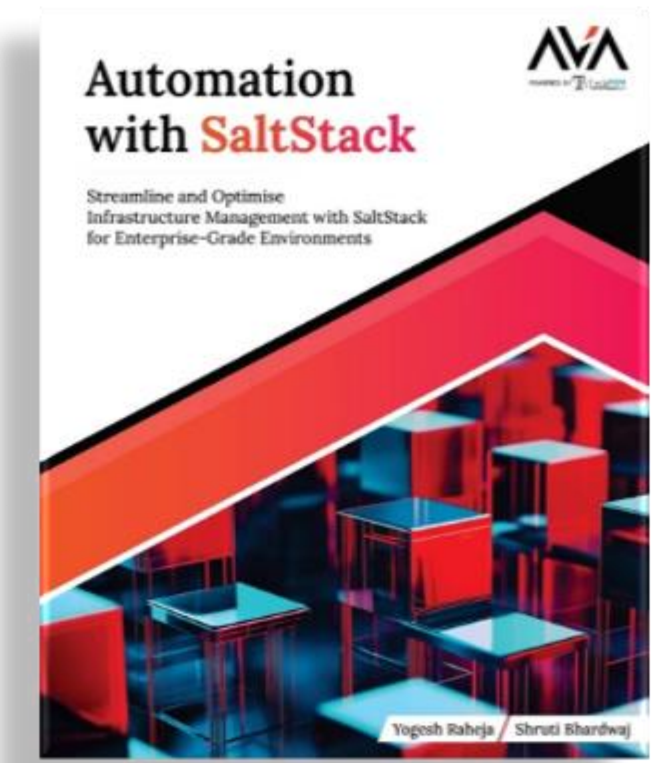
Unlocking Python for the Absolute Beginners - Hands-on

Yogesh Raheja


Effective DevOps with AWS
Second Edition
Implement continuous delivery and integration in the AWS environment
Packt>
Yogesh Raheja, Giuseppe Borgese and Nathaniel Felsen






Automation with SaltStack
Streamline and Optimise Infrastructure Management with SaltStack for Enterprise-Grade Environments
AVA
Yogesh Raheja / Shruti Bhardwaj

Yogesh Raheja

Deepthi Narayan

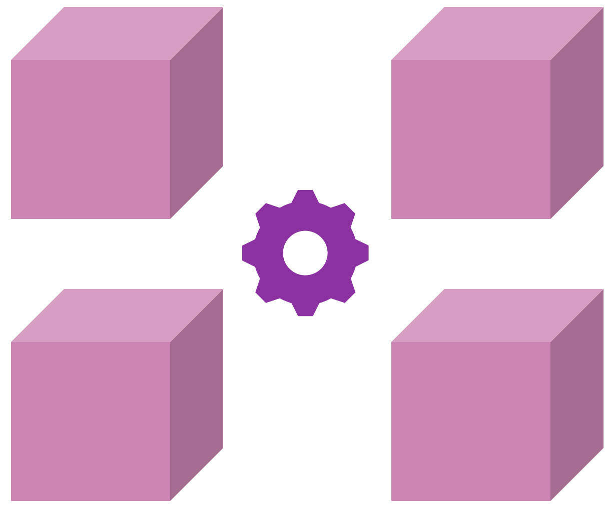
Course Workflow

Images

Containers

Volumes

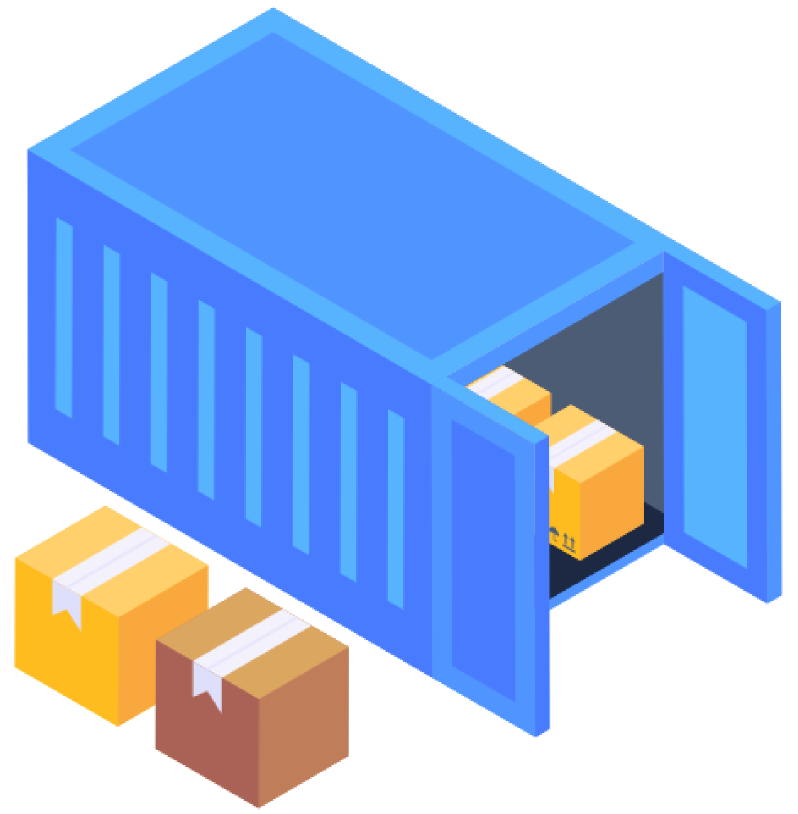
Networks



Course Workflow

- Lectures
- Live Demonstrations
- Assignments

Course Objective



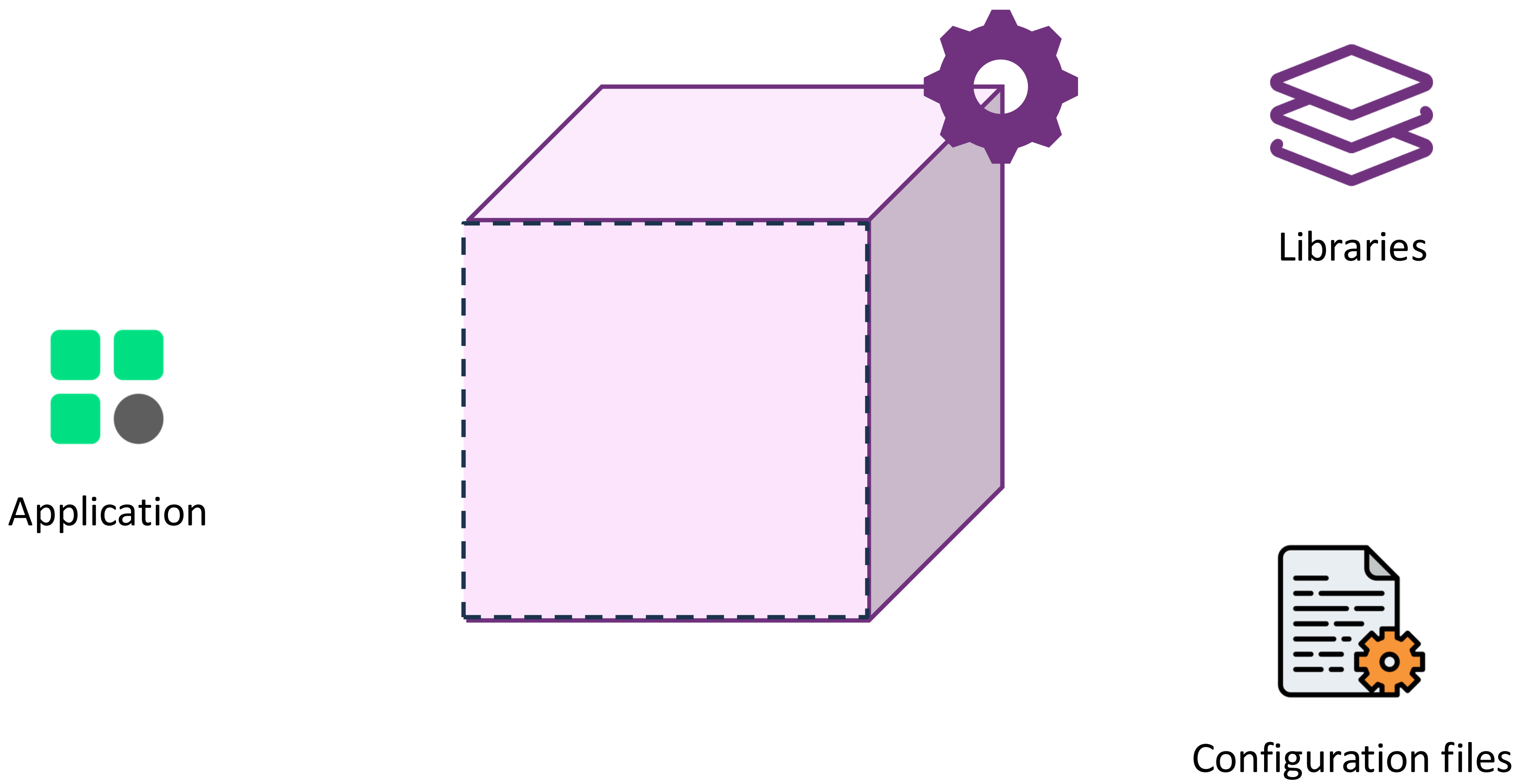
- Podman Introduction
- Setting Up Podman (Desktop/CLI)
- Podman CLI
- Images, Containers, Networks & Volumes
- Pods Lifecycle & Operations
- Image layers & Registries
- Custom Container Images
- Containerize a Python Application
- Multi-stage Build

Podman | **Introduction to Podman**

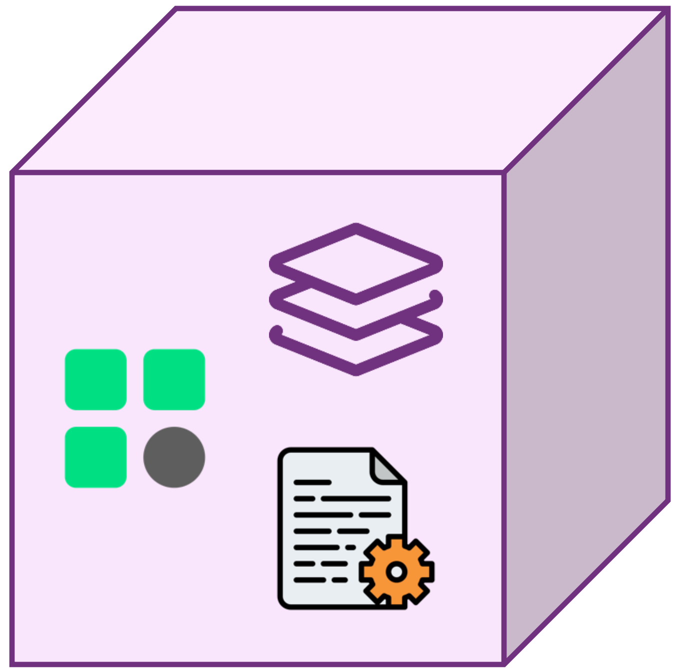
Introduction to Podman

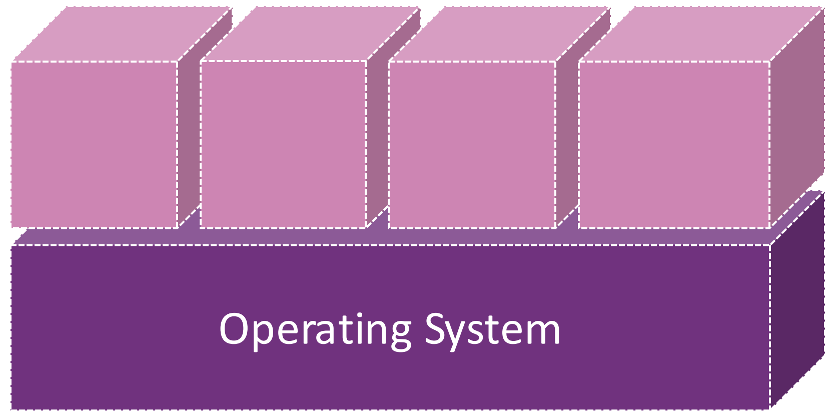
- Fundamentals of Containerization
- Features, Architecture, and Building blocks of Podman
- Podman vs Docker
- Podman Documentation Walkthrough

Introduction to Containerization



Introduction to Containerization





Advantages of Containerization

Environment consistency

Faster scalability

Isolation

Resource efficiency

Development speed

Portability

Agility

Quick startup

Advantages of Containerization

Efficiency **Speed** **Consistency**

Containerization Concepts

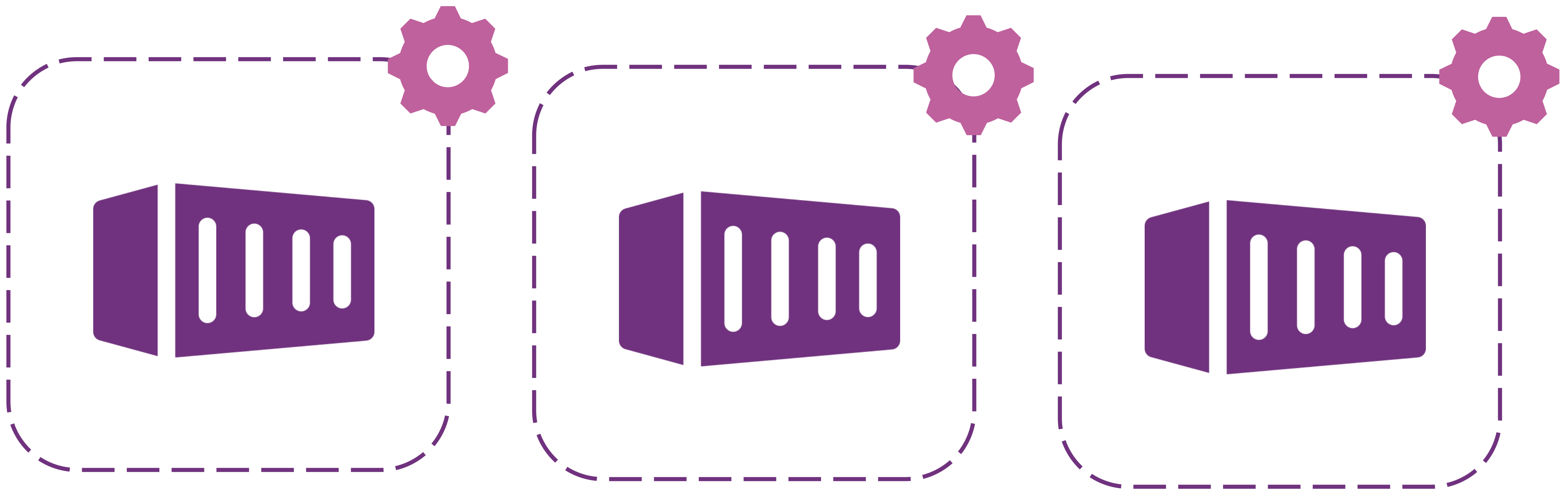
namespaces

cgroups

Containerization Concepts

namespaces

cgroups

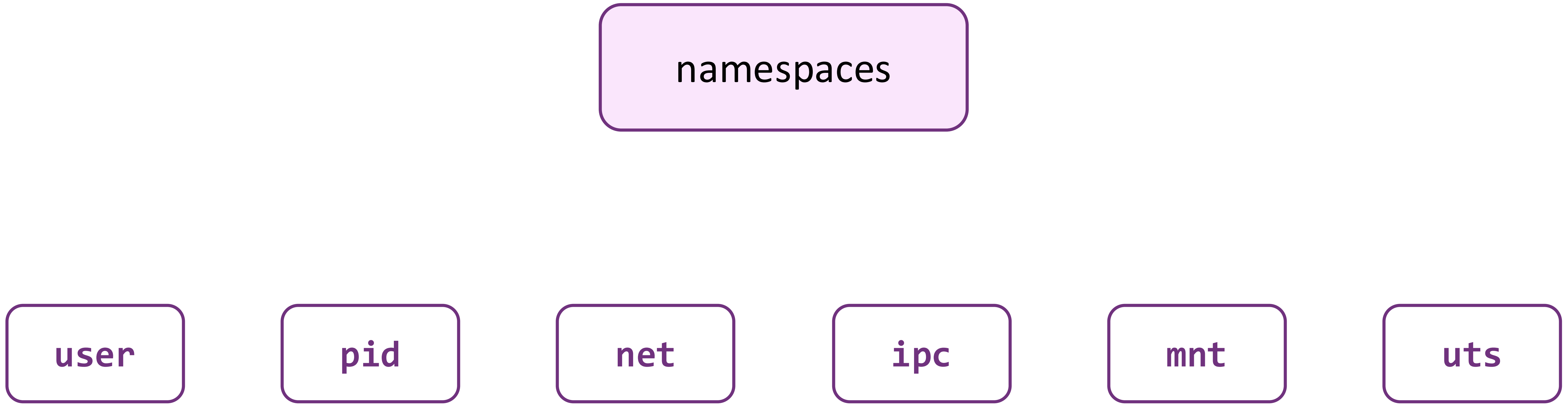


Containerization Concepts

namespaces

- ✓ Isolate and virtualize system resources
- ✓ Ensure each process accesses unique set of resources

Containerization Concepts

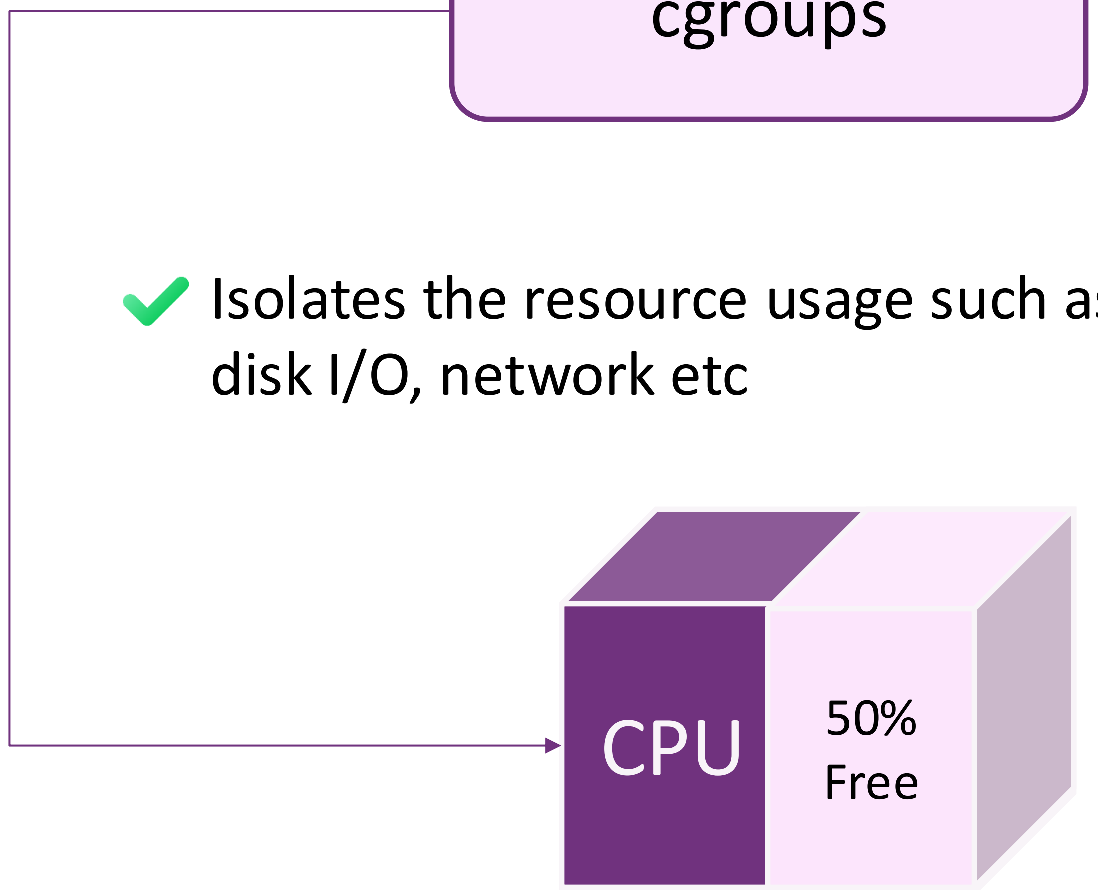


Containerization Concepts

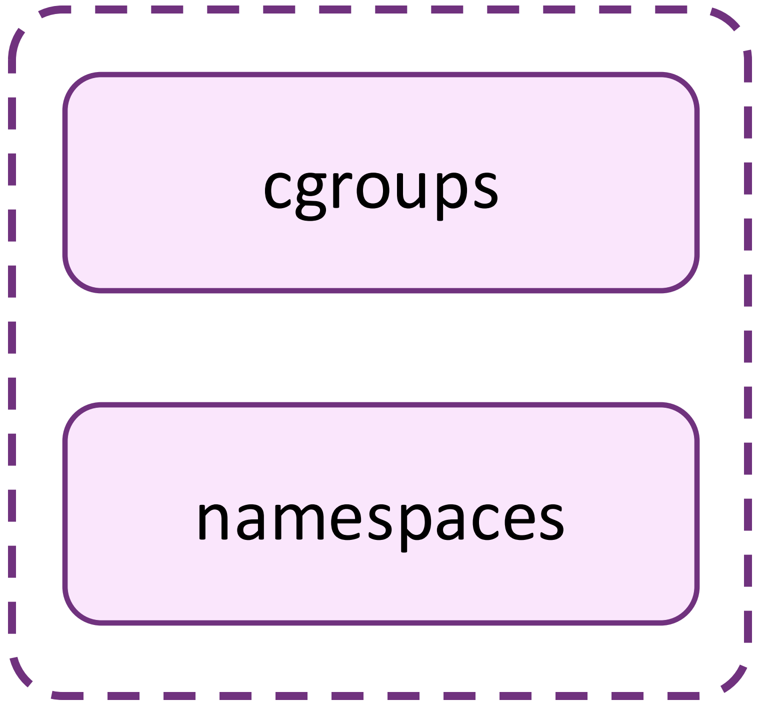
Control groups

cgroups

✓ Isolates the resource usage such as CPU, memory, disk I/O, network etc



Containerization Concepts



✓ Resource management

✓ Isolation and control for processes

Podman Basics

podman

Pod Manager

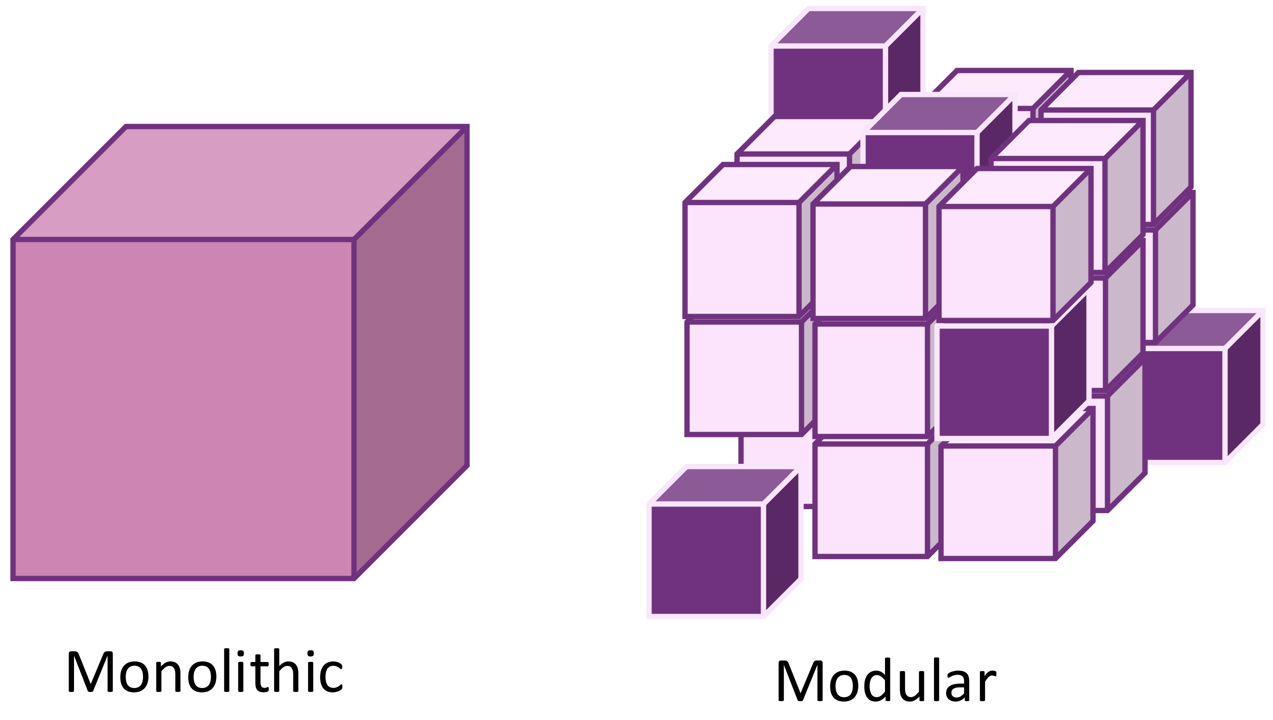
- ✓ Powerful daemon-less
- ✓ Open-source tool
- ✓ Container development, management, and execution

libpod

Podman Basics

podman

Modular Architecture



- Podman reduces resource consumption, making it suitable for environments with limited resources
- Modular approach simplifies troubleshooting and enhances security by limiting the attack surface

buildah

skopeo

Podman Basics

Portability and Flexibility

OPEN CONTAINER INITIATIVE

runc

crun

runV

Podman Basics

✓ Consistent experience across different platforms

Podman Basics

- Buildah creates container images from scratch or existing ones
- Offers fine control over image creation
- No need for a full container runtime

- Skopeo simplifies transferring container images between registries
- Supports registries like Docker Hub and Quay.io
- Enables flexible image storage.

Podman Basics

Rootless Containers

- Rootless containers add significant security by preventing privilege escalation during a breach
- Even if a container engine is compromised, attackers cannot gain root access to the host

Podman Basics

Podman Desktop makes Podman user-friendly for everyone, whether they prefer command-line tools or a visual interface

Podman Basics

Modular Architecture

Rootless Containers

podman

OPEN CONTAINER INITIATIVE

buildah

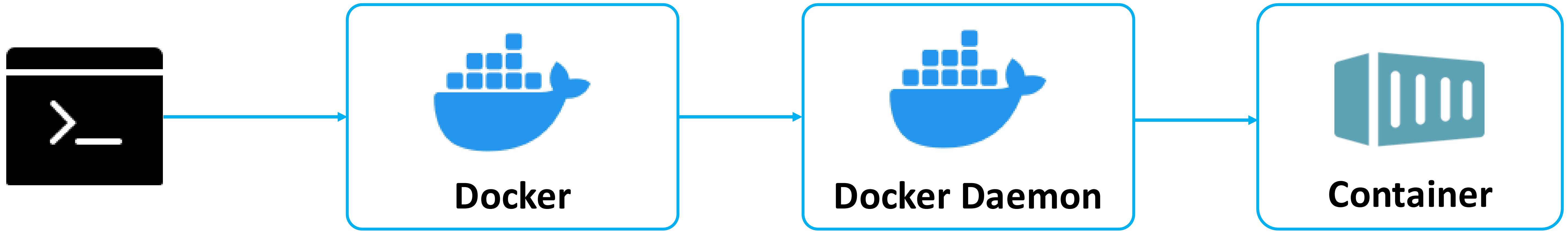
skopeo

Podman Architecture

- Powerful container management tool
- Daemonless
- Enhanced Security and Usability

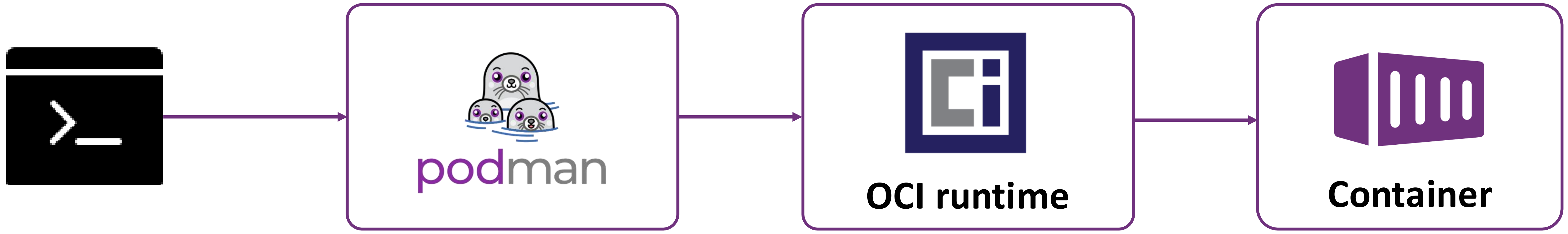
Daemonless operation

Podman Architecture



- Security risks
- Daemons can be exploited by attackers to take control of the host system

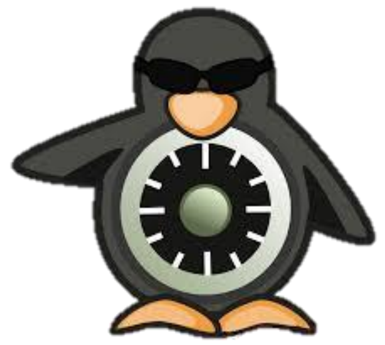
Podman Architecture



- Reduces the attack surface
- Enhanced overall security

Podman Architecture

Container

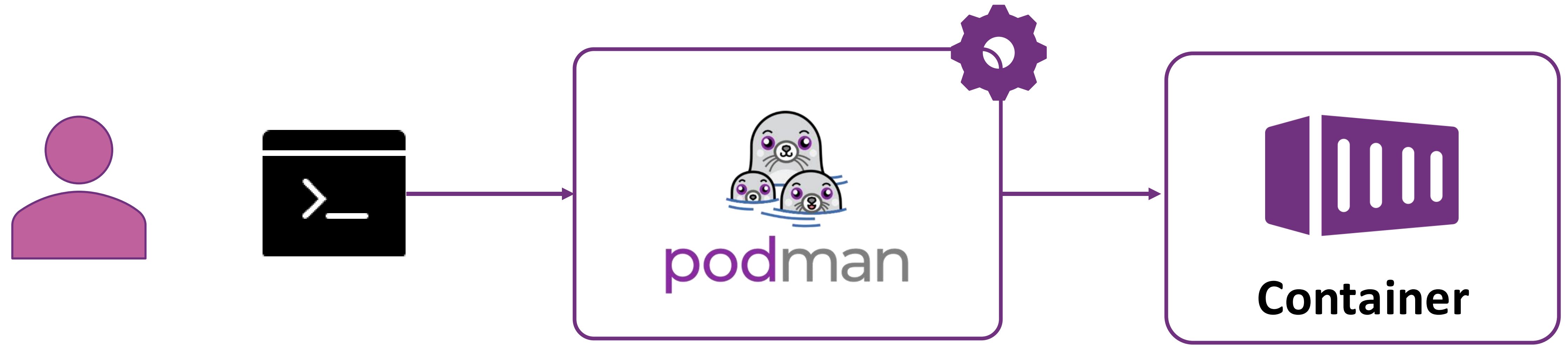


Rootless Containers

- Users can create, run, and manage containers without admin privileges
- Isolated environment -> SELinux labels
- A compromised container cannot access the host system or other containers, enhancing overall system integrity

Podman Architecture

fork/exec model



Enhance reliability by eliminating the single point of failure

Podman Architecture

podman

systemd

- Podman uses systemd for container management without a dedicated daemon
- This integration allows automatic container startup and management with system services
- Ensures container persistence and monitoring with daemonless security

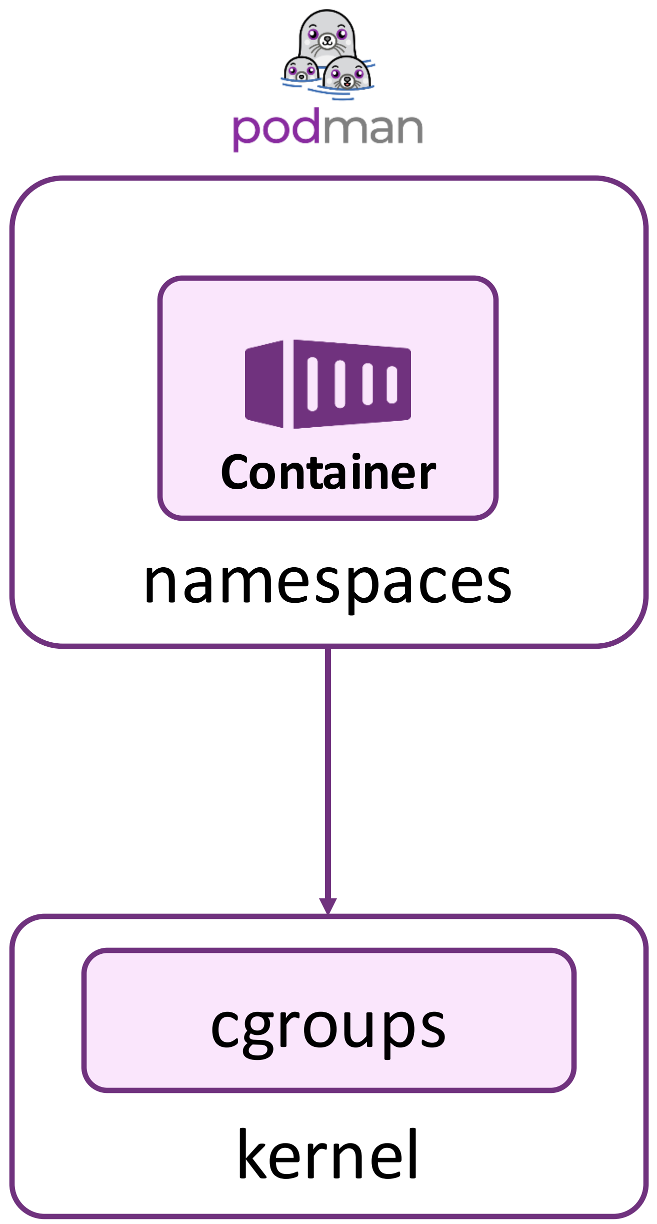
Podman Architecture

- Programmatic container management
- REST (Representational State Transfer)
- Automation and integration

Podman Architecture

- Empowering users
- Maintains separate sets of containers and images -> users can work concurrently on the same host

Podman Architecture



- Resource management and isolation capabilities
- Podman automatically sets up necessary cgroups and namespaces

Podman Building blocks

podman

libpod

Image

Container

Volume

Network

Pod

Podman Building blocks

Image **Container** **Volume** **Network** **Pod**

Image serves as a pre-built template for creating containers

- ✓ Pull
- ✓ Tag
- ✓ Build
- ✓ Share

Podman Building blocks

Image **Container** **Volume** **Network** **Pod**

Container is an instance of an image that runs an application

- ✓ Create
- ✓ Start
- ✓ Stop
- ✓ Manage

Podman Building blocks

Image
Container
Volume
Network
Pod

- Podman volumes enable data persistence beyond a container's lifecycle
- Volumes can be easily attached to containers for reliable storage management

Podman Building blocks

Image
Container
Volume
Network
Pod

- Podman provides network management to facilitate communication between containers
- You can utilize Podman’s default network configuration for easy external connectivity

Podman Building blocks

Image
Container
Volume
Network
Pod

- Podman supports pods, enabling multiple containers to run together and share resources

Podman vs Docker

podman

docker®

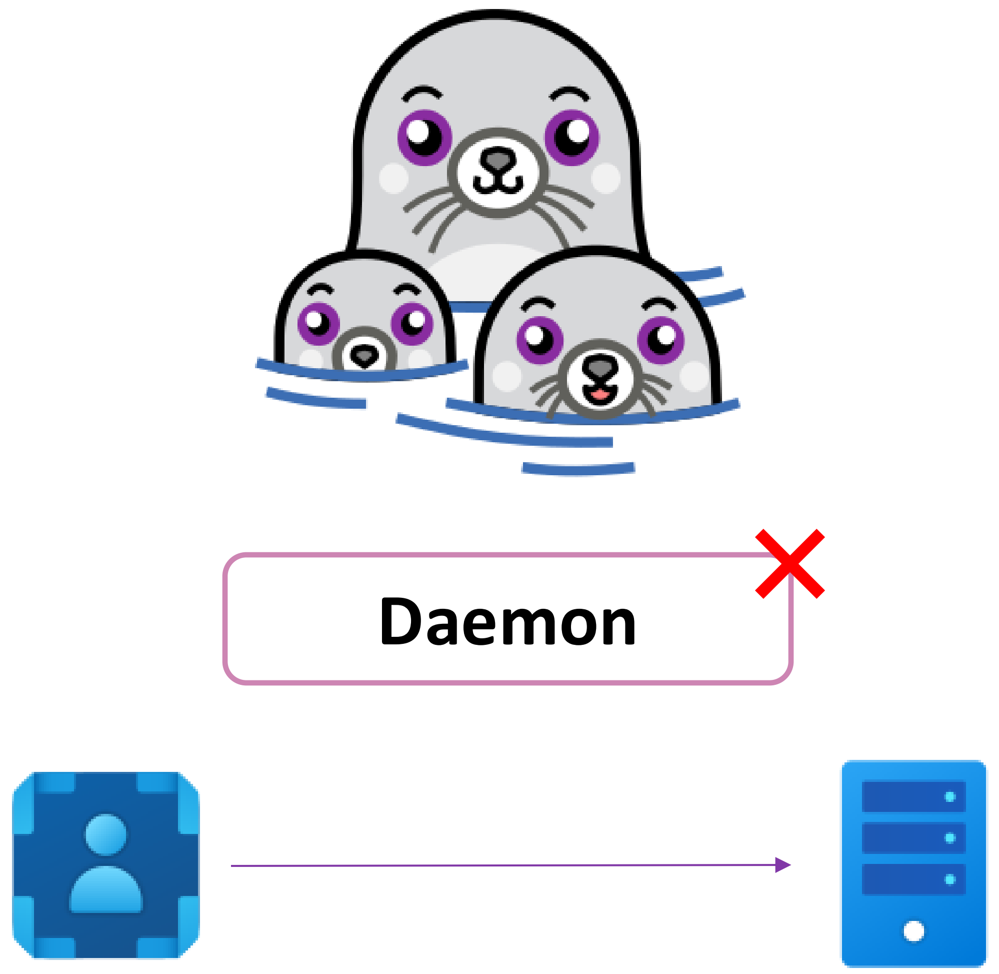
cri-o

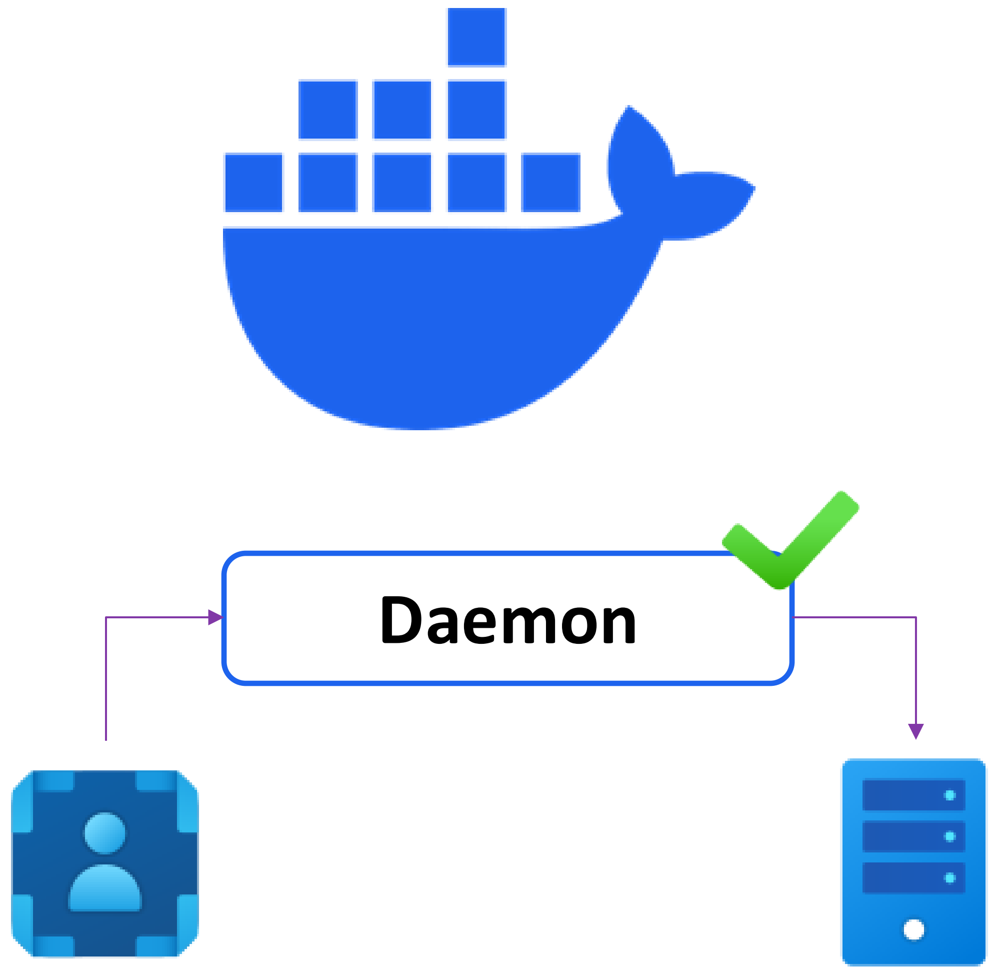
containerd

Podman vs Docker

- Podman and Docker provide intuitive command-line interfaces for managing containers
- alias docker = podman

1. Architecture and Management





This approach enables quick interactions and faster management of containers

This can add some complexity since the daemon must be running to manage your containers

2. Security Features

Single point of failure

Daemon

It supports rootless containers, enabling non-privileged users to run them safely

Docker now has a rootless mode, allowing users to run containers securely without root privileges

3. Performance and Startup Time

Fast Container Startup

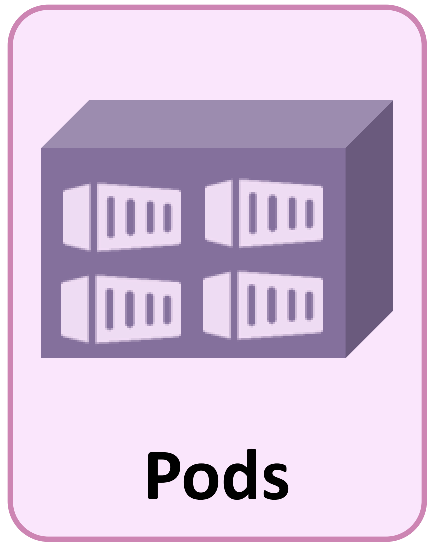
Takes a bit longer to start containers

4. Image Building and Support

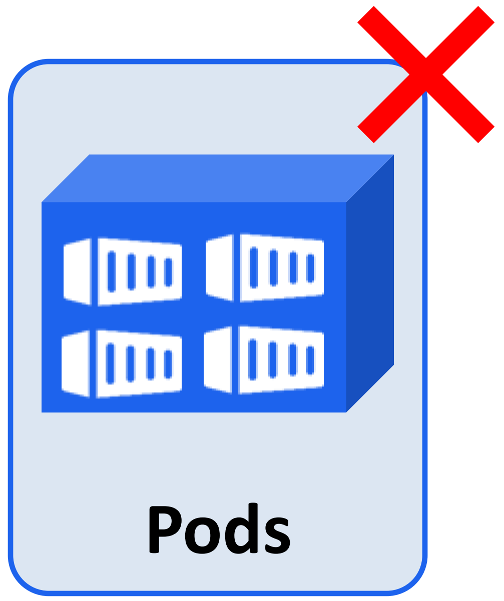
Lightweight and efficient builds without running daemon

Includes built-in image building but adds overhead and reduces flexibility

5. Pod Concept



Lightweight groups where multiple containers can share container namespaces



Docker lacks pod functionality

Demonstration | Podman Documentation Walkthrough

Summary

- ❖ Introduction to Containerization
- ❖ Podman: Features and Daemonless Architecture
- ❖ Core Building Blocks of Podman
- ❖ Podman vs Docker: Key Differences
- ❖ How to Navigate Podman Documentation

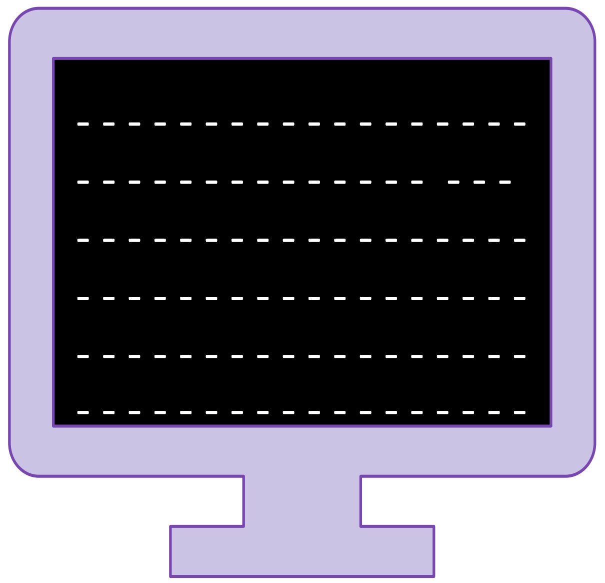
Podman | **Getting Started with Podman**

Getting Started with Podman

- Installation of Podman on Linux, macOS and Windows
- Podman CLI
- Hands-on demonstrations

Podman Installation Methods

Podman CLI



- Command-line utility
- Advanced tasks
- Terminal commands
- Text-based interface

Podman Desktop

- Graphical user interface
- Beginners
- Visual approach
- Extensions

Native CLI (Podman Machine)

- **How to install Podman CLI on a Linux Ubuntu machine**
- **How to install Podman Desktop on both Windows and Mac**

Podman Installation Methods

Linux Ubuntu (Podman CLI)	Mac (Podman Desktop)	Windows (Podman Desktop)
Ubuntu 20.10 and newer	Intel or Apple M1 (for .dmg file) Others: Use the universal binary	6 GB RAM Virtual Machine Platform Enabled WSL2 enabled - Requirements to enable WSL2: - Windows 10 (Build 19043+) or Windows 11, 64-bit - Admin access - If on a VM: Nested Virtualization enabled

Podman Installation Methods

Linux Ubuntu (Podman CLI)	Mac (Podman Desktop)	Windows (Podman Desktop)
Ubuntu 20.10 and newer	Intel or Apple M1 (for .dmg file) Others: Use the universal binary	6 GB RAM Virtual Machine Platform Enabled WSL2 enabled - Requirements to enable WSL2: - Windows 10 (Build 19043+) or Windows 11, 64-bit - Admin access - If on a VM: Nested Virtualization enabled

Demonstration | Installing Podman on Linux

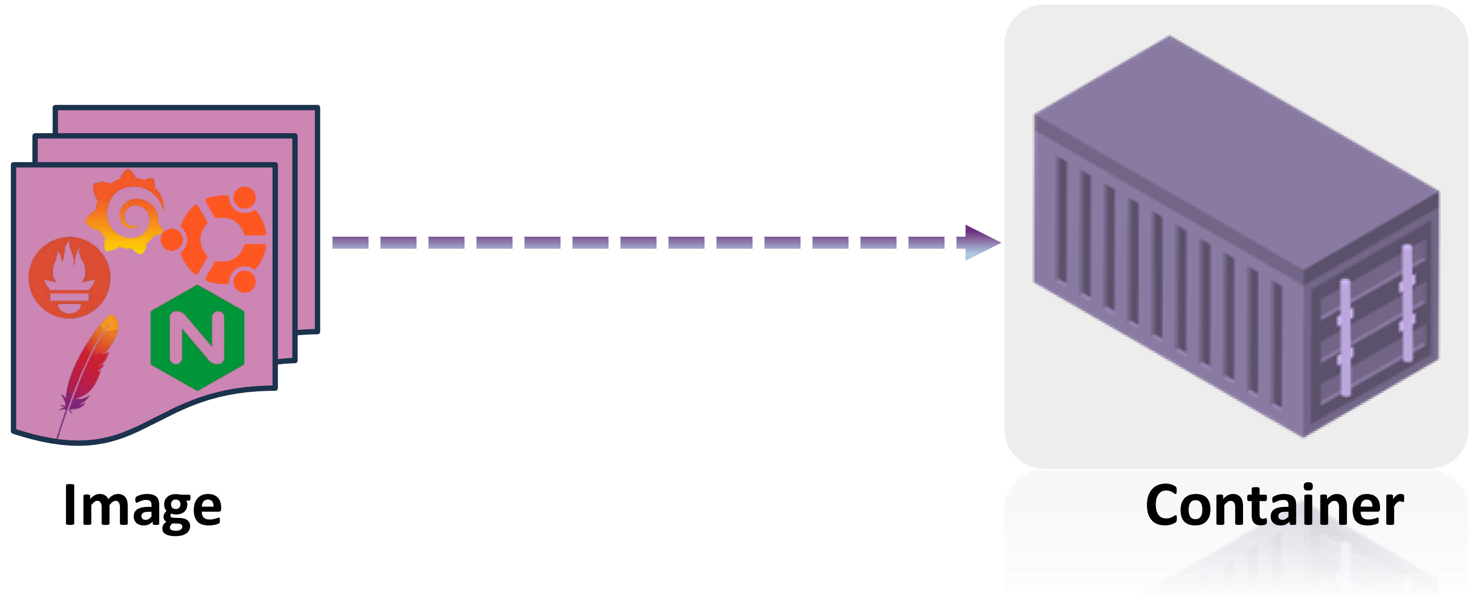
Demonstration | Introduction to Podman Commands

Demonstration | Podman Desktop on Mac

Summary

- ❖ Podman Installation Methods
- ❖ Podman CLI on Linux
- ❖ Podman Desktop on macOS and Windows
- ❖ Podman CLI and its basic usage

Container Images



- *Read-only templates*

Container Images

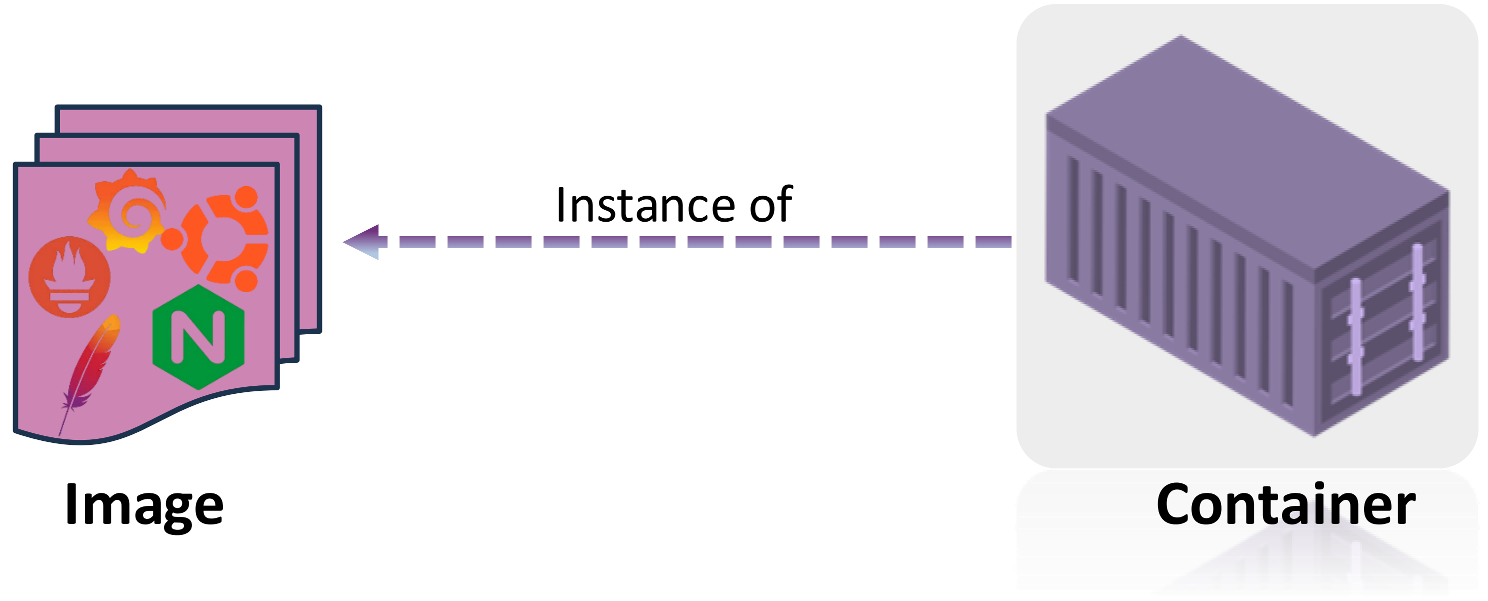
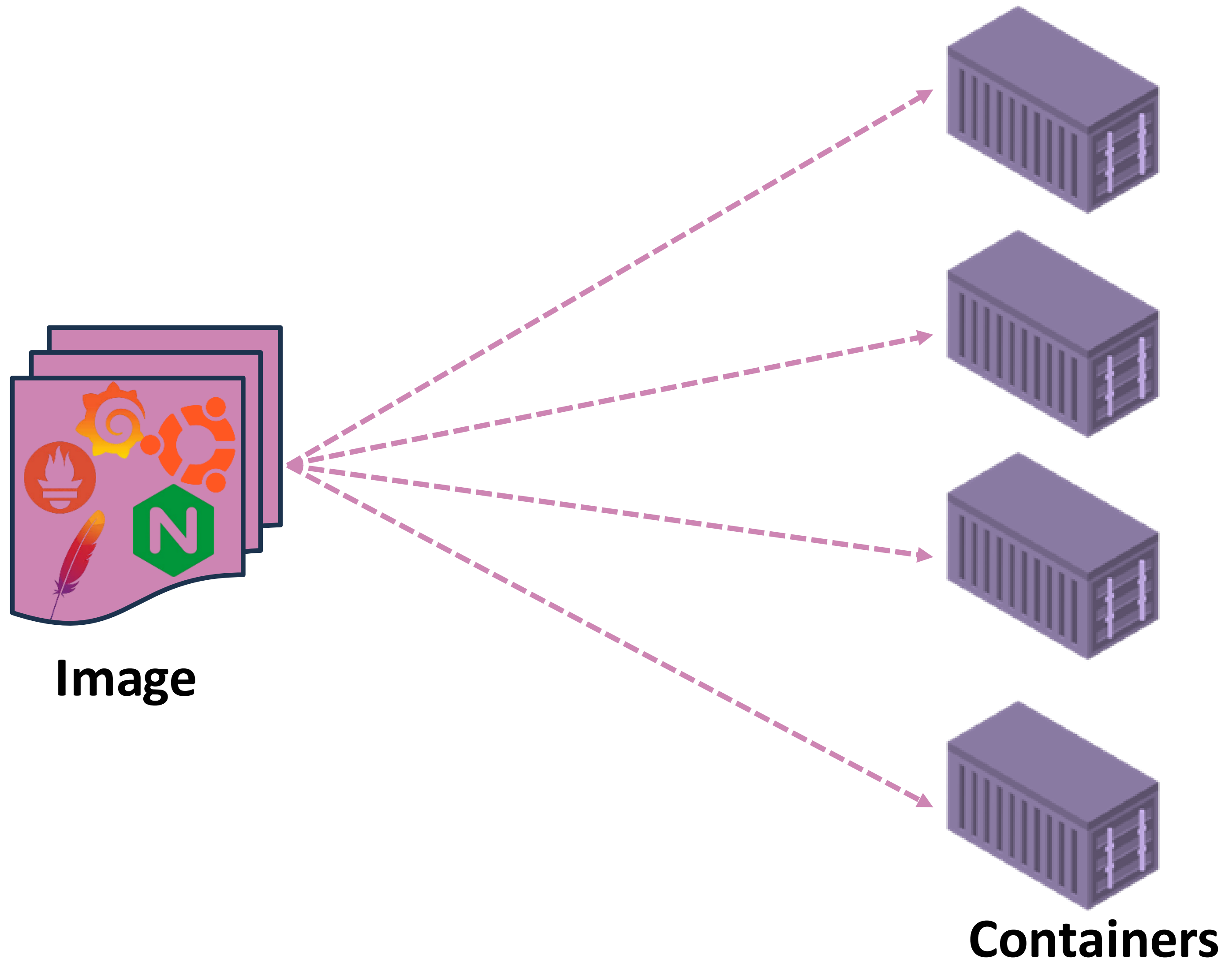


Image = Class

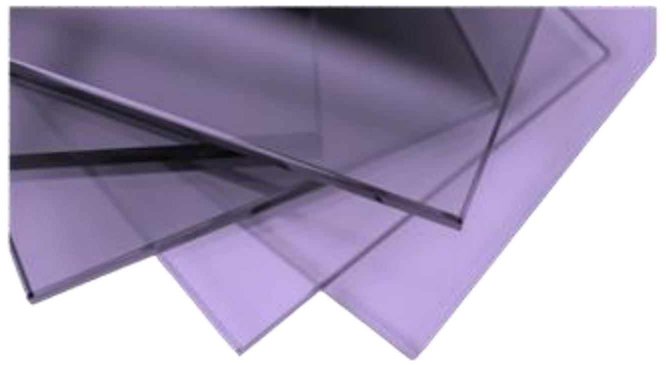
Container = Object

- *Environment and Dependencies*
  - ✓ *Code*
  - ✓ *Libraries*
  - ✓ *Runtime*
  - ✓ *Configurations*

Container Images



Container Images



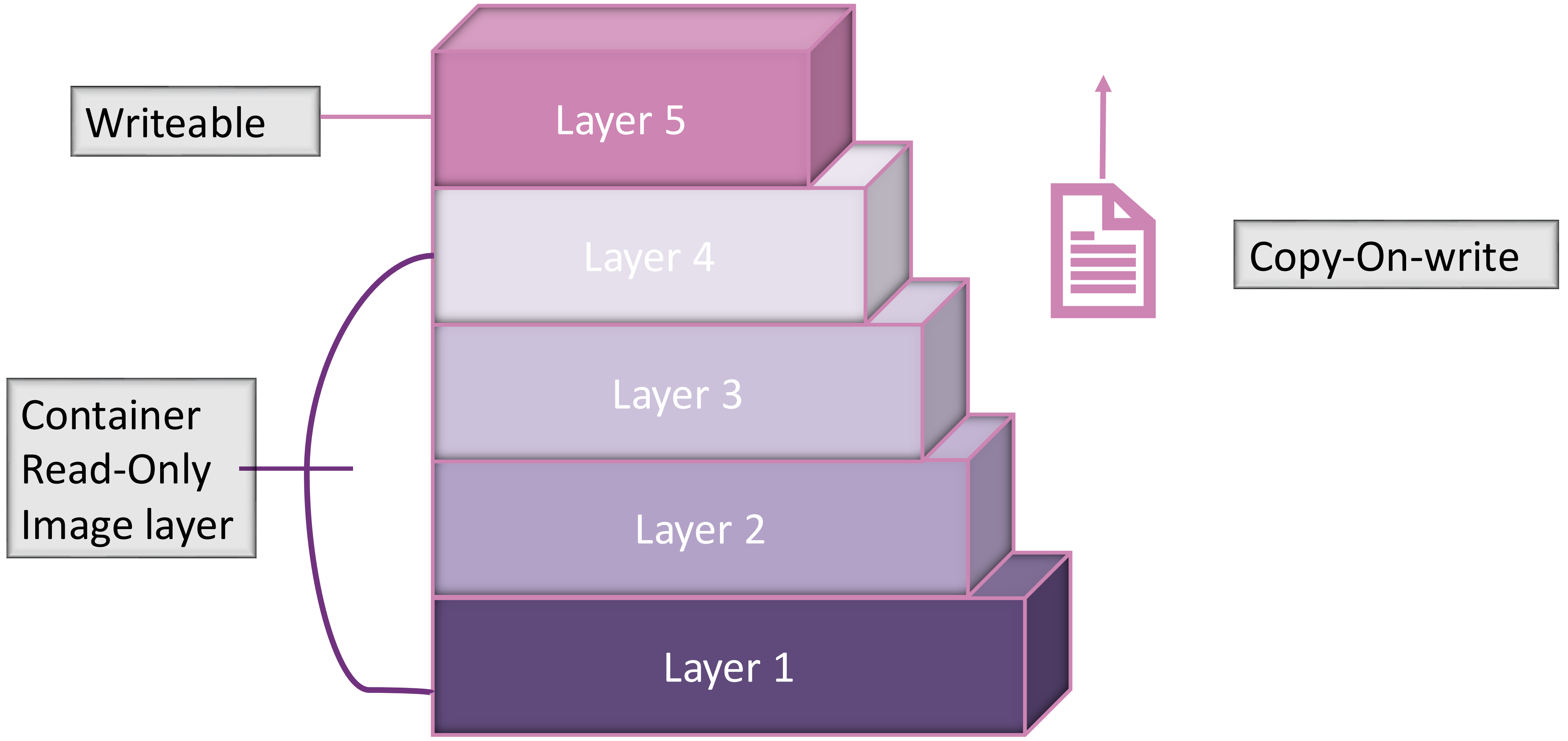
Files

Libraries

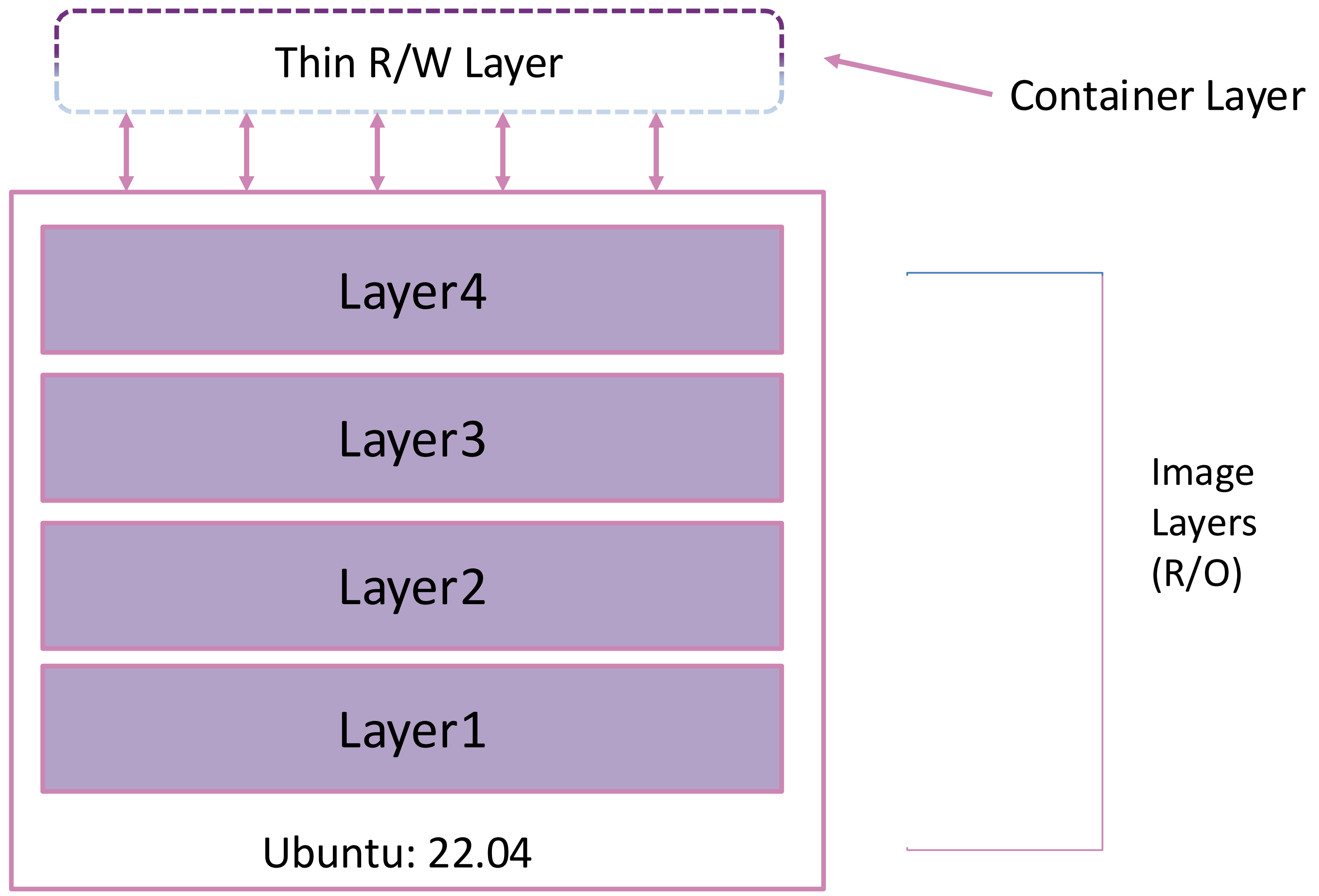
Configurations

Container Images

Union File System (Native Overlay)

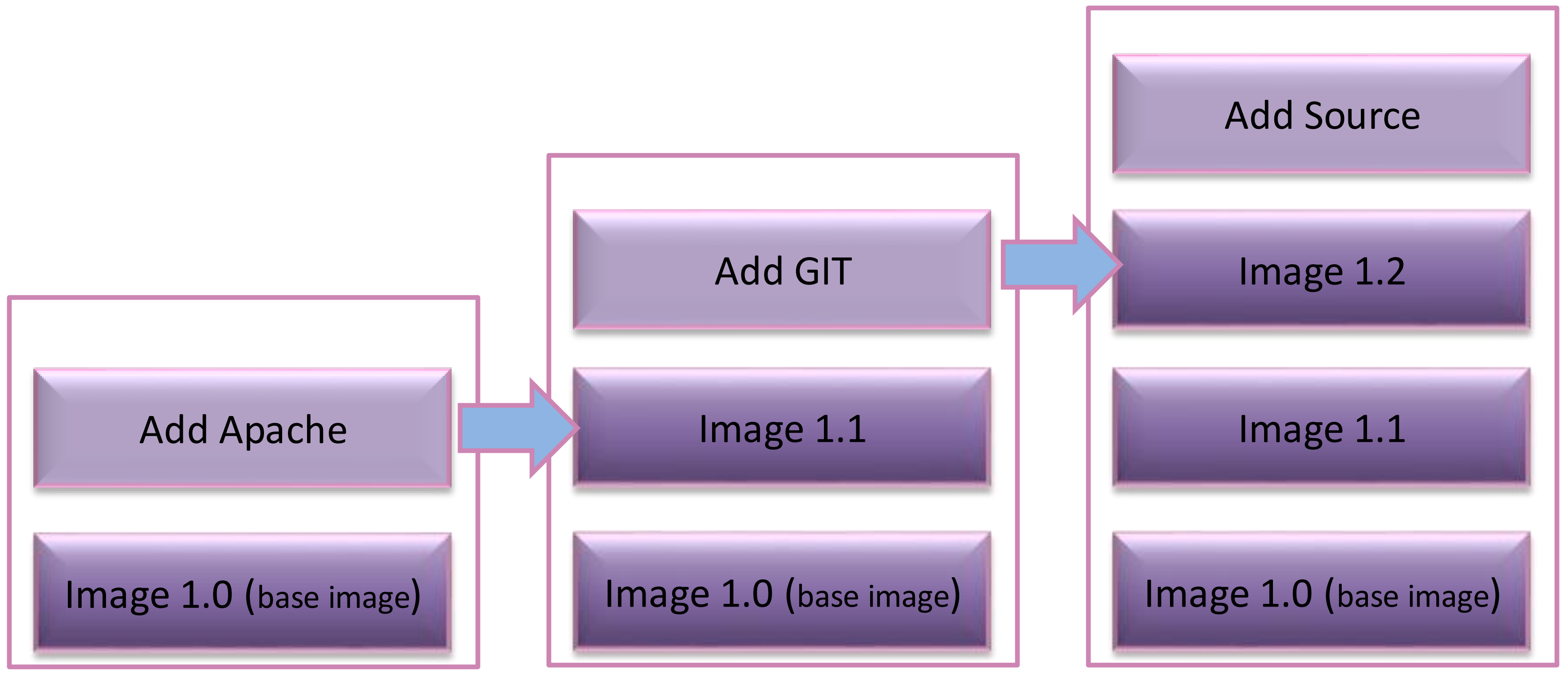


Container Images



Container Images

- ✓ Gradual modifications
- ✓ Reusability



Container Images

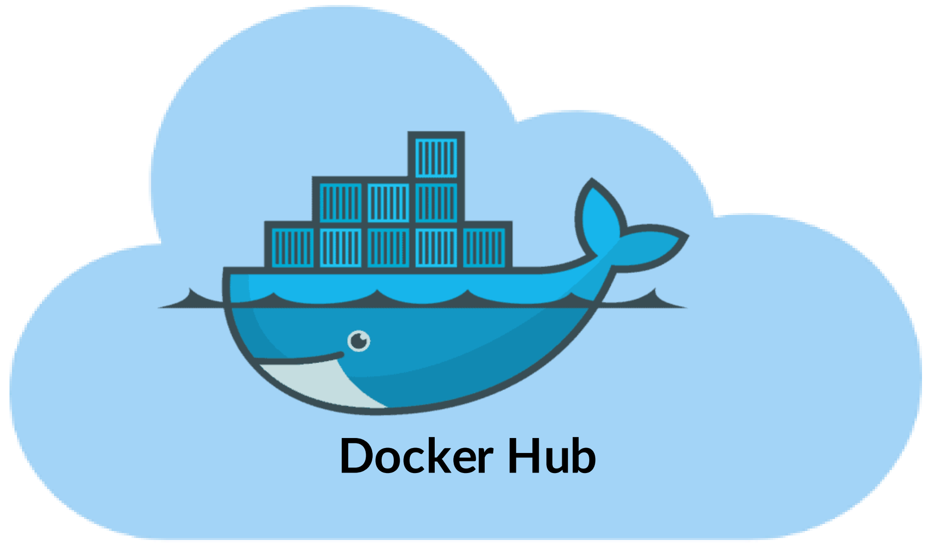
Containerfile

```
FROM ubuntu:24.04
RUN  apt update
RUN  apt install -y apache2
COPY ./index.html /var/www/html/index.html
EXPOSE 80
CMD  ["apache2ctl", "-D", "FOREGROUND"]
```

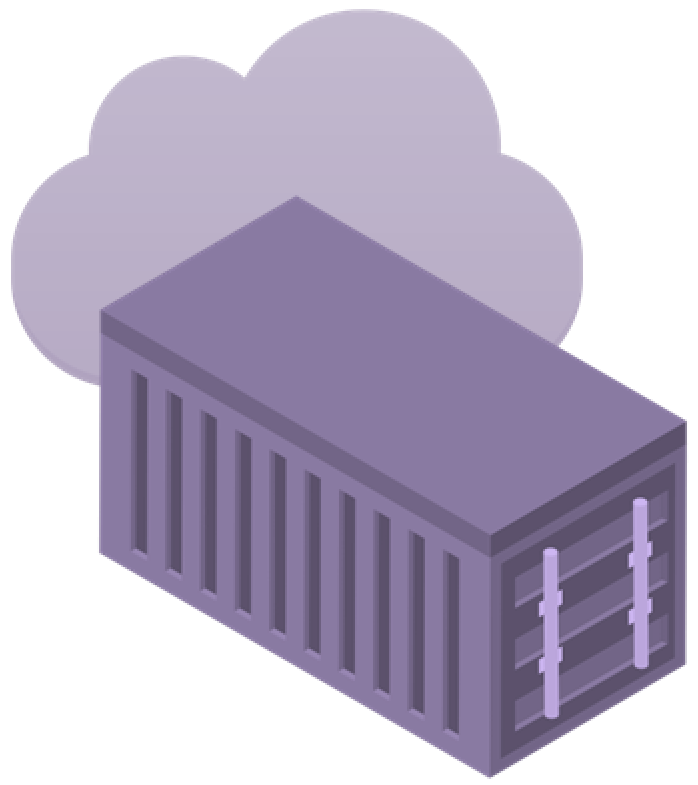
Registries

Private

Public



Container Registries: Docker Hub and Quay.io



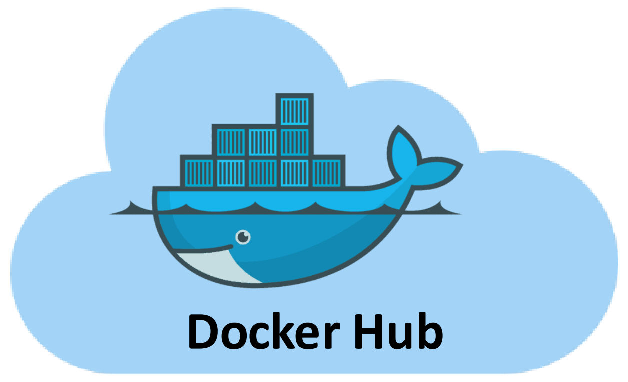
Container Registry

- *Store, share and distribute container images*

Registry

- *Docker Hub*
- *Quay.io*
- *Harbor*
- *JFrog Artifactory*
- *GitHub Packages*

Container Registry: Docker Hub



- ✓ *Repository of container images*
- ✓ *Find, share, and distribute images*
- ✓ *Public and Private*
- ✓ *Docker Inc*
- ✓ *Large community*
- ✓ *Simple, user-friendly interface*

Container Registry: Docker Hub

Docker Hub

- Web Interface
- Integration Options
- Collection of Images, extensions and Plugins
- Image Tags (Versions)
- Collaboration
- Automated Builds

Container Registry: Docker Hub

Types of Images

Official

Verified Publishers

Sponsored OSS

User

deepthianarayan/podman

thinknyx/jenkins

Container Registry: Quay.io

Quay.io

- ✓ *Robust container registry*
- ✓ *Public and Private*
- ✓ *Automated builds*
- ✓ *Security*
- ✓ *Quay Enterprise (by Red Hat)*
- ✓ *OCI compliance*

Demonstration | Docker Hub

Managing images with Podman CLI

Description	Command	Alternative command
Download an image	podman pull <image_name> podman pull <registry_name>/<repository_name>/<image_name>:tag	podman image pull <image_name> podman image pull <registry_name>/<repository_name>/ <image_name>:tag
List all the images	podman images	podman image list (or) podman image ls
Assign an additional image name	podman tag <source_image> <target_image>	podman image tag <source_image> <target_image>
Show the history of an image	podman history <image_name>	podman image history <image_name>
Display detailed information of an image	podman inspect <image_name>	podman image inspect <image_name>
Remove an image	podman rmi <image_name>	podman image rm <image_name>

Demonstration | Managing images with Podman CLI

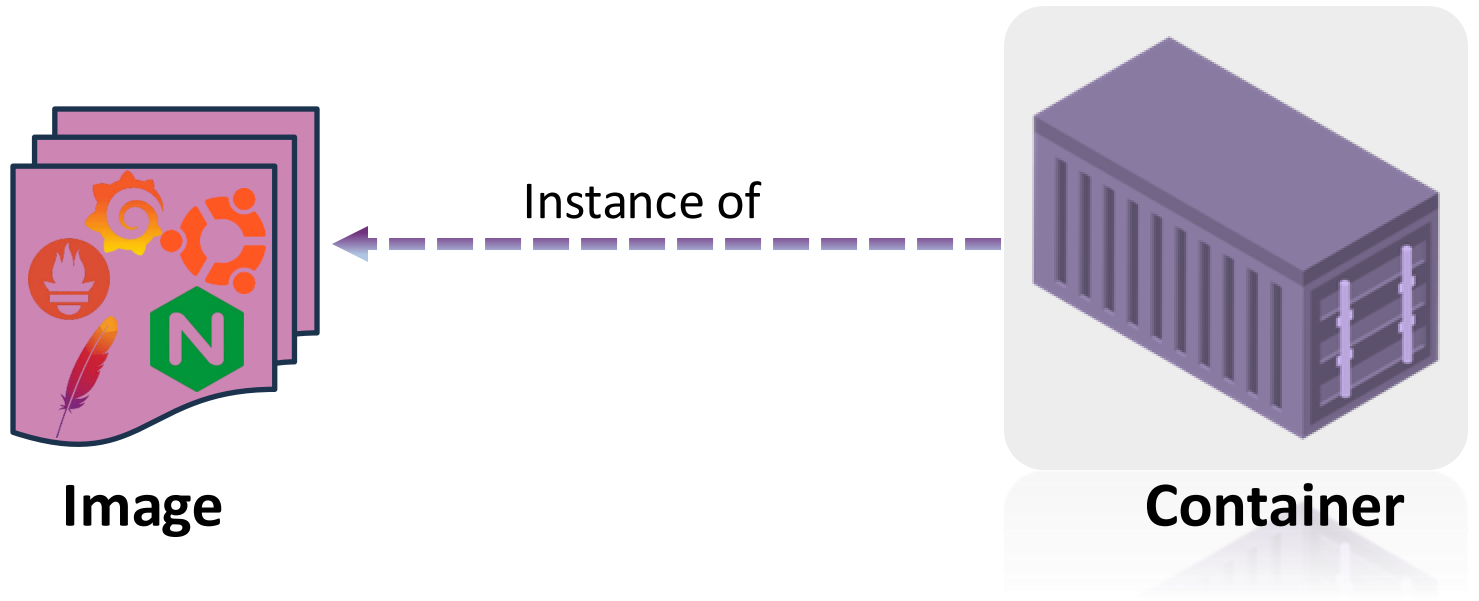
Summary

- Container images
- Various Registries Options (Docker Hub/Quay.io)
- Image operations using Podman CLI

Podman | **Working with Containers in Podman**

Containers in Podman

- ✓ Code
- ✓ Libraries
- ✓ Runtime
- ✓ Configurations

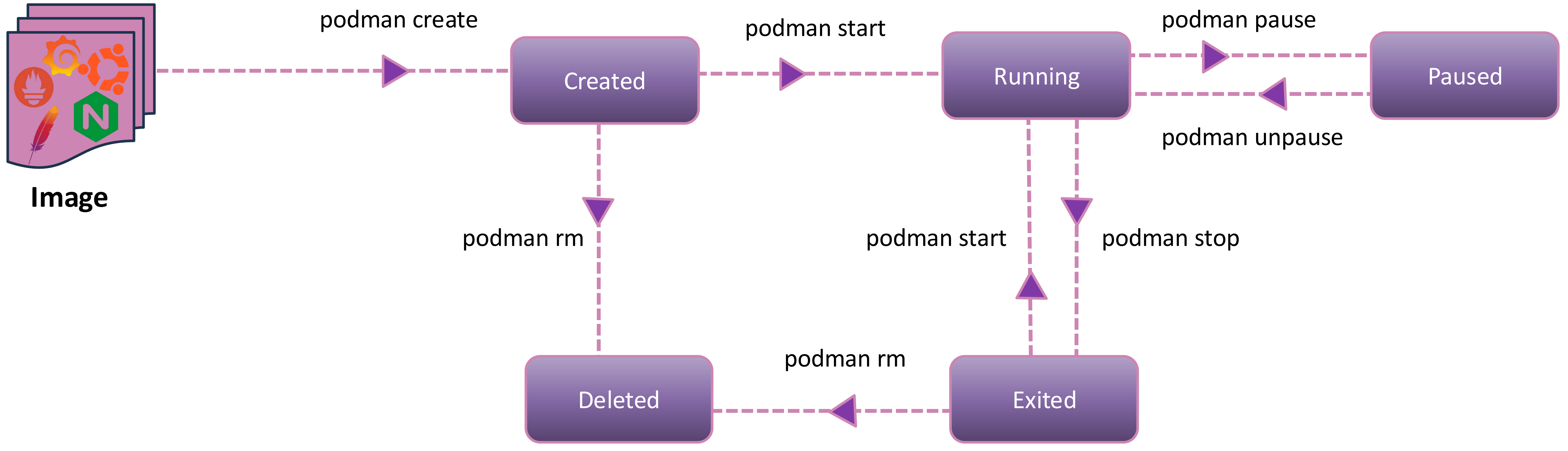


- ✓ Read-only templates

```
podman run <image_name>
```

```
podman run alpine
```

Containers in Podman



Managing Containers with Podman CLI

Description	Command	Alternative command
Run a container	podman run <image_name>	podman container run <image_name>
List all the containers	podman ps -a	podman container ps -a(or) podman container list -a(or) podman container ls -a
Execute a command in a running container	podman exec <container_name> [ARG]	podman container exec <container_name> [ARG]
Rename a container	podman rename <old_container_name> <new_container_name>	podman container rename <old_container_name> <new_container_name>
View the logs of a container	podman logs <container_name>	podman container logs <container_name>
View the detailed information about a container	podman inspect <container_name>	podman container inspect <container_name>

Managing Containers with Podman CLI

Description	Command	Alternative command
Stop a container	podman stop <container_name>	podman container stop <container_name>
Start a container	podman start <container_name>	podman container start <container_name>
Remove a Container	podman rm <container_name>	podman container rm <container_name>

Demonstration | Managing Containers with Podman CLI

Demonstration | Rootless Containers

Summary

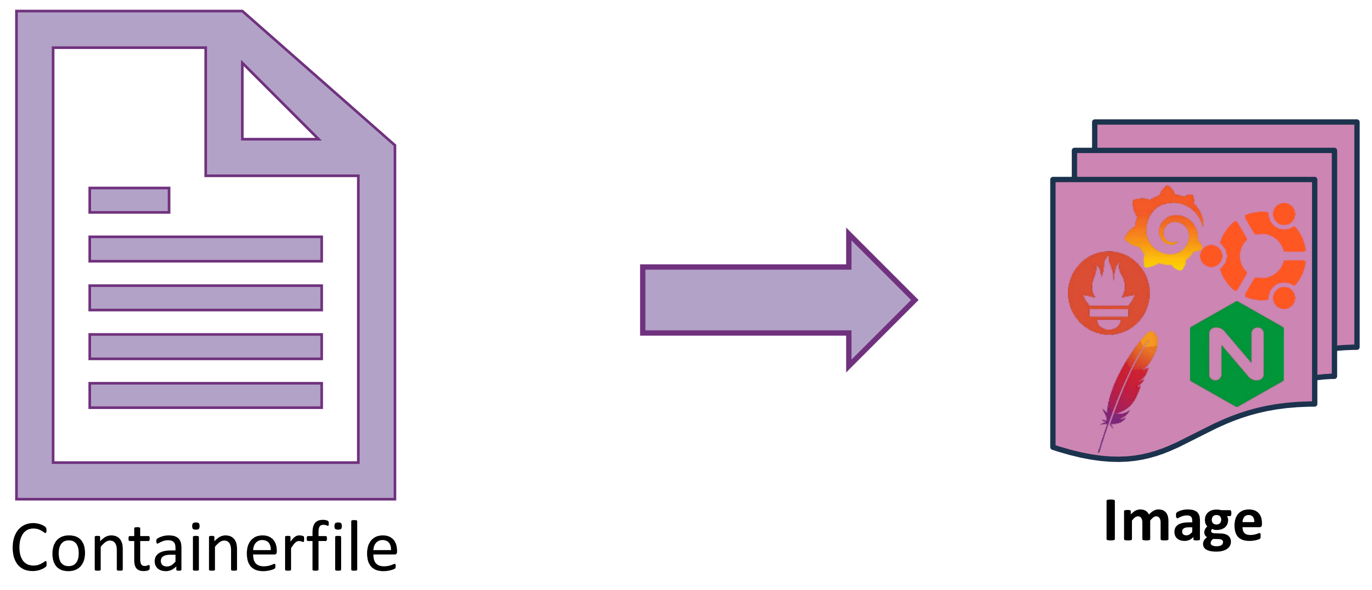
- Container lifecycle management using Podman CLI

Podman | **Building images with Containerfile**

Building images with Containerfile

- Containerfile Fundamentals
- Developing and validating Containerfile
- Building custom container images

Understanding Containerfile



- Script with instructions to construct the image

Understanding Containerfile

Containerfile

Steps to construct Container image

- Setting up environment
- Installing dependencies
- Copying application files

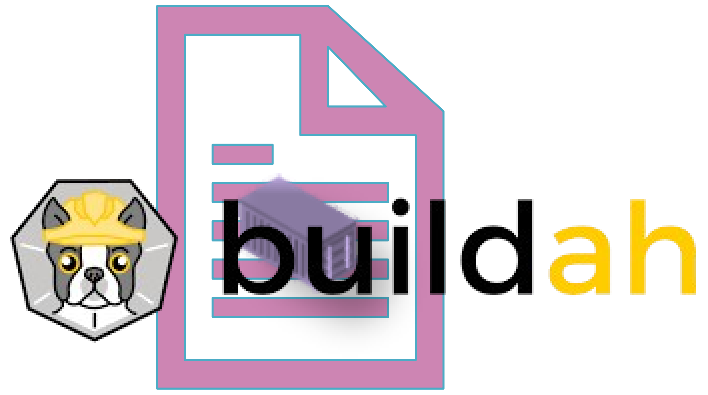
Build process

Images are consistent, reproducible, and ready to deploy across any environment

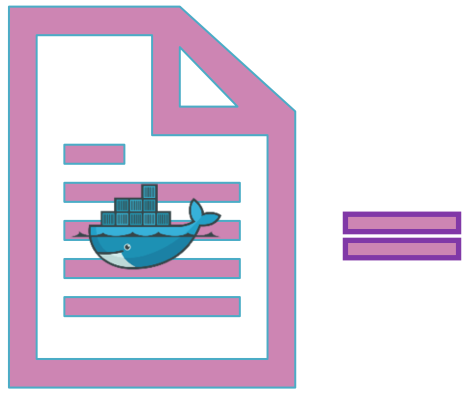
Understanding Containerfile

INSTRUCTION arguments

```
FROM busybox
```



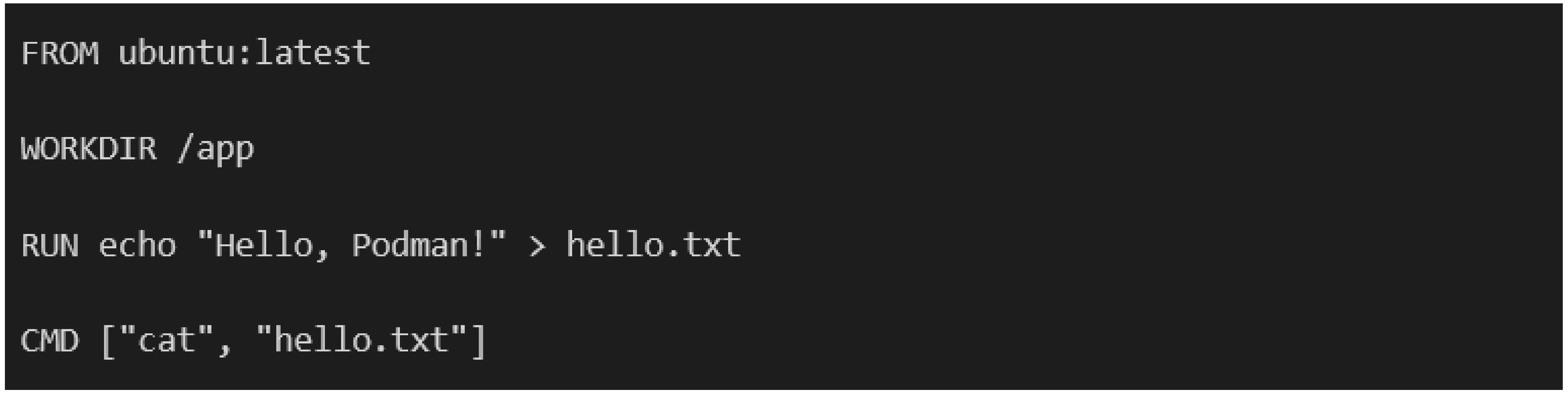
Containerfile



Dockerfile

Instruction	Description
ADD	Add local or remote files and directories.
ARG	Use build-time variables.
CMD	Specify default commands.
COPY	Copy files and directories.
ENTRYPOINT	Specify default executable.
ENV	Set environment variables.
EXPOSE	Describe which ports your application is listening on.
FROM	Create a new build stage from a base image.
LABEL	Add metadata to an image.
MAINTAINER	Specify the author of an image.
ONBUILD	Specify instructions for when the image is used in a build.
RUN	Execute build commands.
USER	Set user and group ID.
VOLUME	Create volume mounts.
WORKDIR	Change working directory.

Understanding Containerfile



```
FROM ubuntu:latest

WORKDIR /app

RUN echo "Hello, Podman!" > hello.txt

CMD ["cat", "hello.txt"]
```

- Containerfile instructions are executed sequentially
- Each instruction translates to an image layer

Understanding Dockerfiles

Description	Command	Alternative command
Build an image from a Containerfile/Dockerfile	podman build -t <image_name> .	podman image build -t <image_name> .

Demonstration | Creating a Dockerfile/Containerfile

Demonstration | Validating Containerfile and building images

Demonstration | Running a Container from our own image

Summary

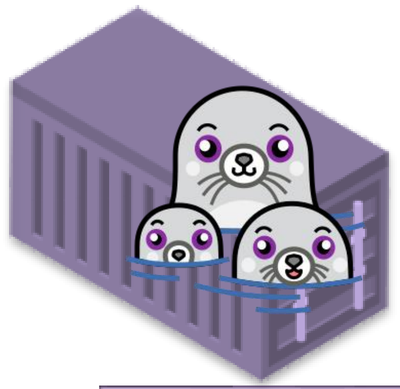
- Created and validated a Containerfile
- Built a custom image
- Created a container using our own image

Podman | **Networking in Podman**

Networking in Podman

- Networking in Podman for rootful and rootless containers
- Network management operations using CLI
- Hands-on demonstrations

Networking in Podman



Container

- ❑ Basics of Podman networking
- ❑ Rootful vs Rootless containers

Networking in Podman

Networking Modes

Rootful Containers

- Netavark
- Containers to receive routable IP addresses
- Fully integrate with the host network and external networks

Rootless Containers

- Slirp4netns
- Offers basic networking functionality
- Does not support routable IP addresses

Networking in Podman

Basic network setups

Bridge Networking

- Default network for rootful containers
- Creates a virtual bridge

- ➢ **Podman network create**
  - ✓ bridge, macvlan, and ipvlan

✓ **Rootless users --> default networking mode-->slirp4netns**

✓ **Podman 4.0--> rootless users--> netavark = rootful setup**

Macvlan Networking

- Direct access to physical network interface on host
- Each container can obtain its own IP address

Managing networks with Podman Commands

Description	Command	Alternative command
List all the networks	podman network ls	-
Create a network	podman network create <network_name>	-
Display detailed information of a network	podman network inspect <network_name>	-
Remove a network	podman network rm <network_name>	-
Remove all unused networks	podman network prune	-

Demonstration | Networking in Podman

Demonstration | Connecting Containers

Summary

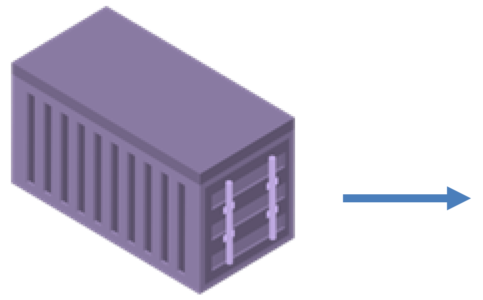
- Fundamentals of networking in Podman
- Essential commands for managing networks
- Demonstration
    - Rootful and Rootless containers

Podman | **Volumes in Podman**

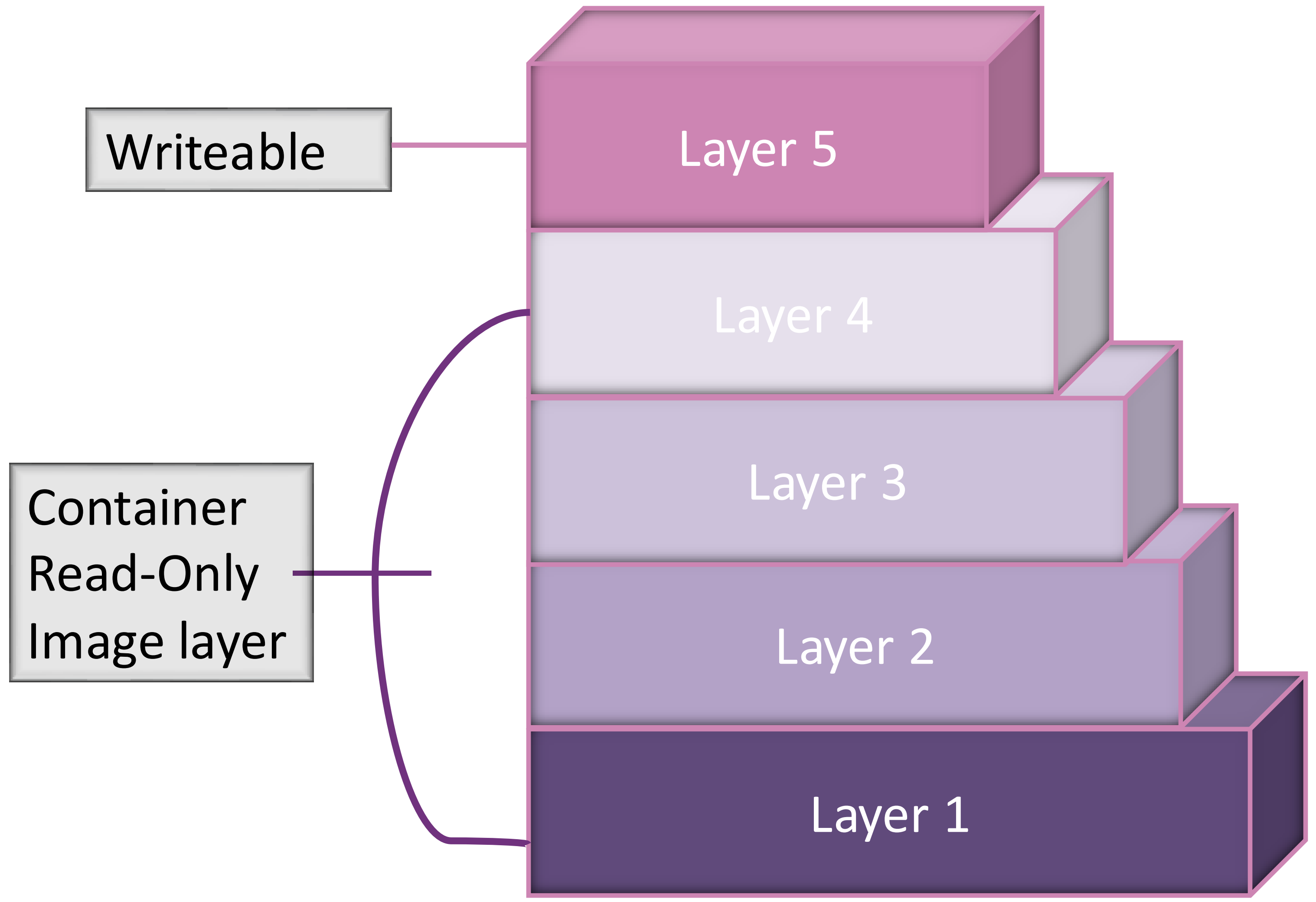
Volumes in Podman

- Data persistence in containers
- Volume management operations using CLI
- Hands-on demonstrations

Data Storage in Containers



Ephemeral and disposable



Data Storage in Containers

Data can be made permanent → Adding that data to the image

- ✗ Not ideal and practical solution
- ✗ Increase in the size of the image
- ✗ Tightly coupled
- ✗ Performance issues

❑ Named Volumes	✓ Independent of container lifecycle
❑ Bind mounts	✓ No increase in the size of the image

Data Storage in Containers

❑ Named Volumes

- ✓ Preferred option
- ✓ Ideal for Critical data storage needs
- ✓ Backup, restore and migrate
- ✓ Long-term data management

❑ Bind mounts

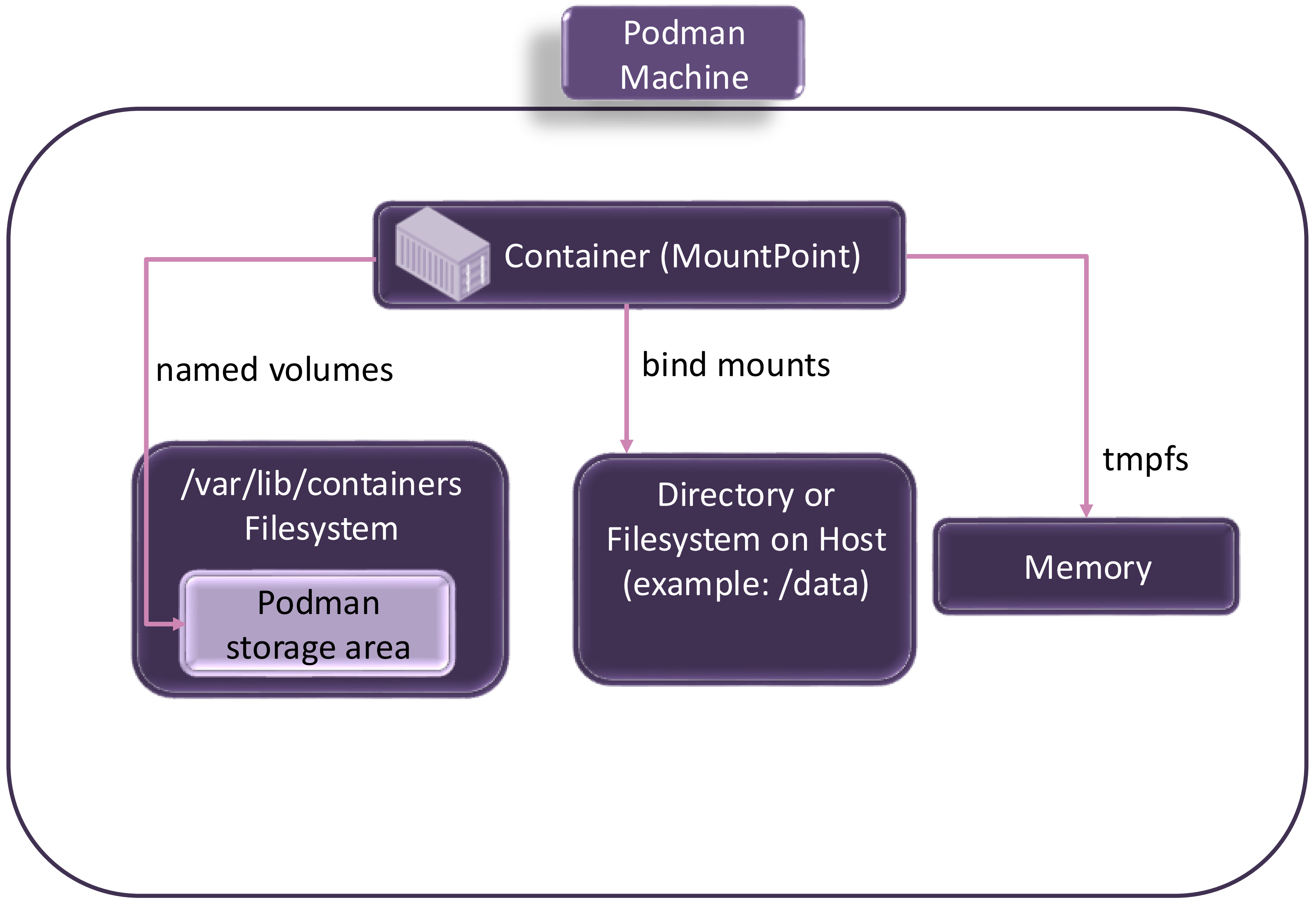
Map a directory or file from the host’s filesystem to container

- ✗ Less flexible and portable
- ✗ Careful configuration
- ✓ Dev environments

❑ tmpfs

tmpfs for Linux and named pipes for windows

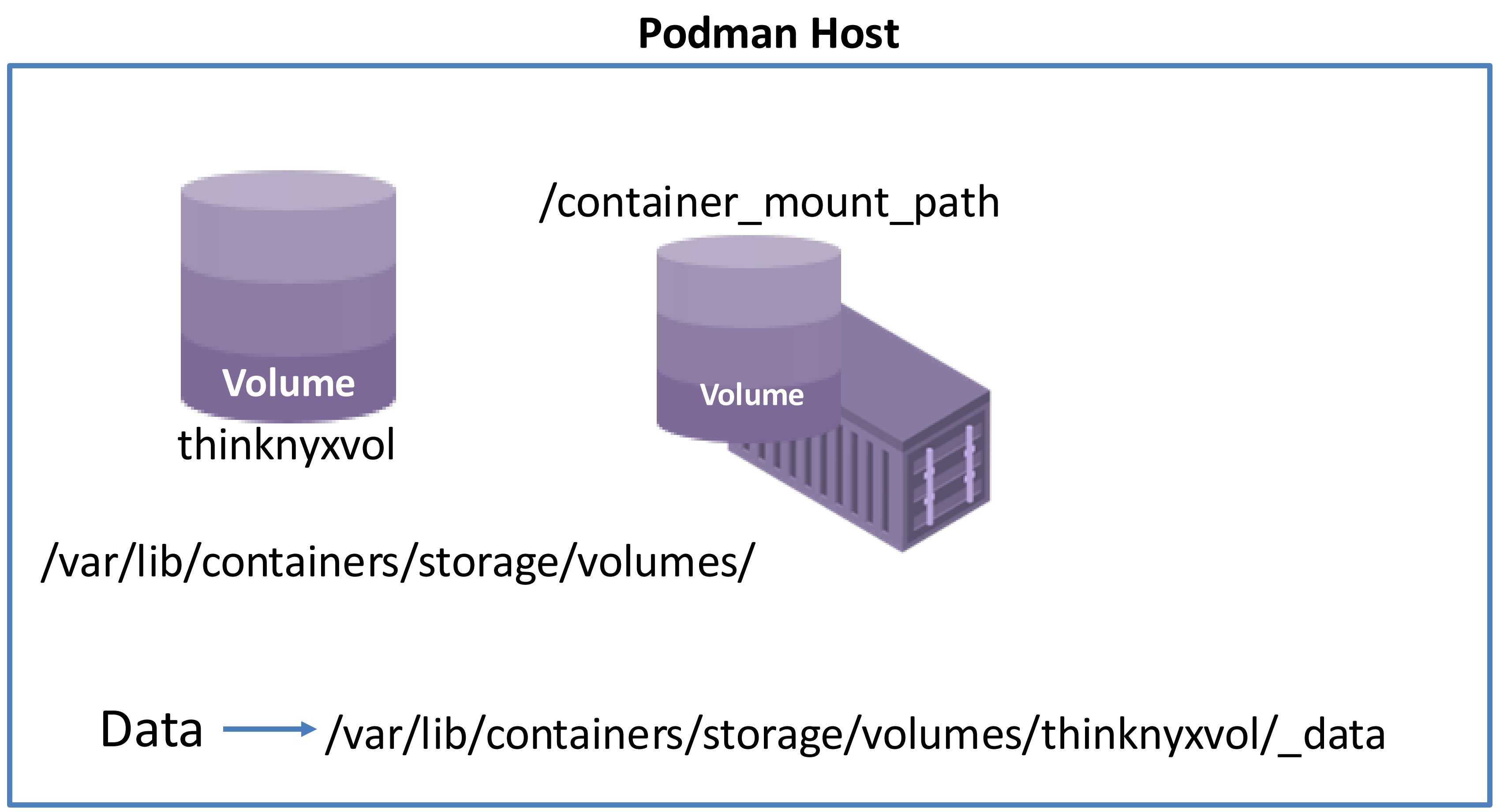
- ✓ in-memory solutions
- ✓ Fast
- ✗ Temporary

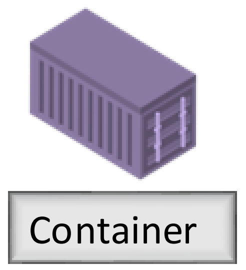


Volumes in Podman

Persist data beyond container lifecycle ✓

Rootful container


Podman Host
/container_mount_path
Volume
thinknyxvol
Volume
/var/lib/containers/storage/volumes/
Data
/var/lib/containers/storage/volumes/thinknyxvol/_data



Default Location = /var/lib/containers/storage/overlay/

Rootless container

Path: /home/thinknyx/.local/share/containers/storage/volumes/

Volumes in Podman

- ❑ **Named Volumes**
- ❑ **Bind Mounts**
- ❑ **Anonymous Volumes**

Volumes in Podman

- ❑ **Named Volumes**
  - User provides specific name
  - Unique within the host
  - Easy to identify
  - Shared with multiple containers
- ❑ Bind Mounts
- ❑ Anonymous Volumes

```
podman volume create thinknyxvol

podman run -d --name=thinknyxcon -p 8081:80 --mount
type=volume,src=thinknyxvol,target=/usr/local/apache2/htdocs httpd:latest
```

Volumes in Podman

- ❑ Named Volumes
  - User provides specific name
  - Unique within the host
  - Easy to identify
  - Shared with multiple containers

- ❑ **Bind Mounts**
  - Mount host directories into containers
  - Enables file sharing between host and container

- ❑ Anonymous Volumes

```
podman run -itd --name=thinknyxcon -p 8081:80 --mount
type=bind,src=/data,target=/usr/local/apache2/htdocs httpd:latest
```

Volumes in Podman

❑ Named Volumes

User provides specific name
Unique within the host
Easy to identify
Shared with multiple containers

❑ Bind Mounts

Mount host directories into containers
Enables file sharing between host and container

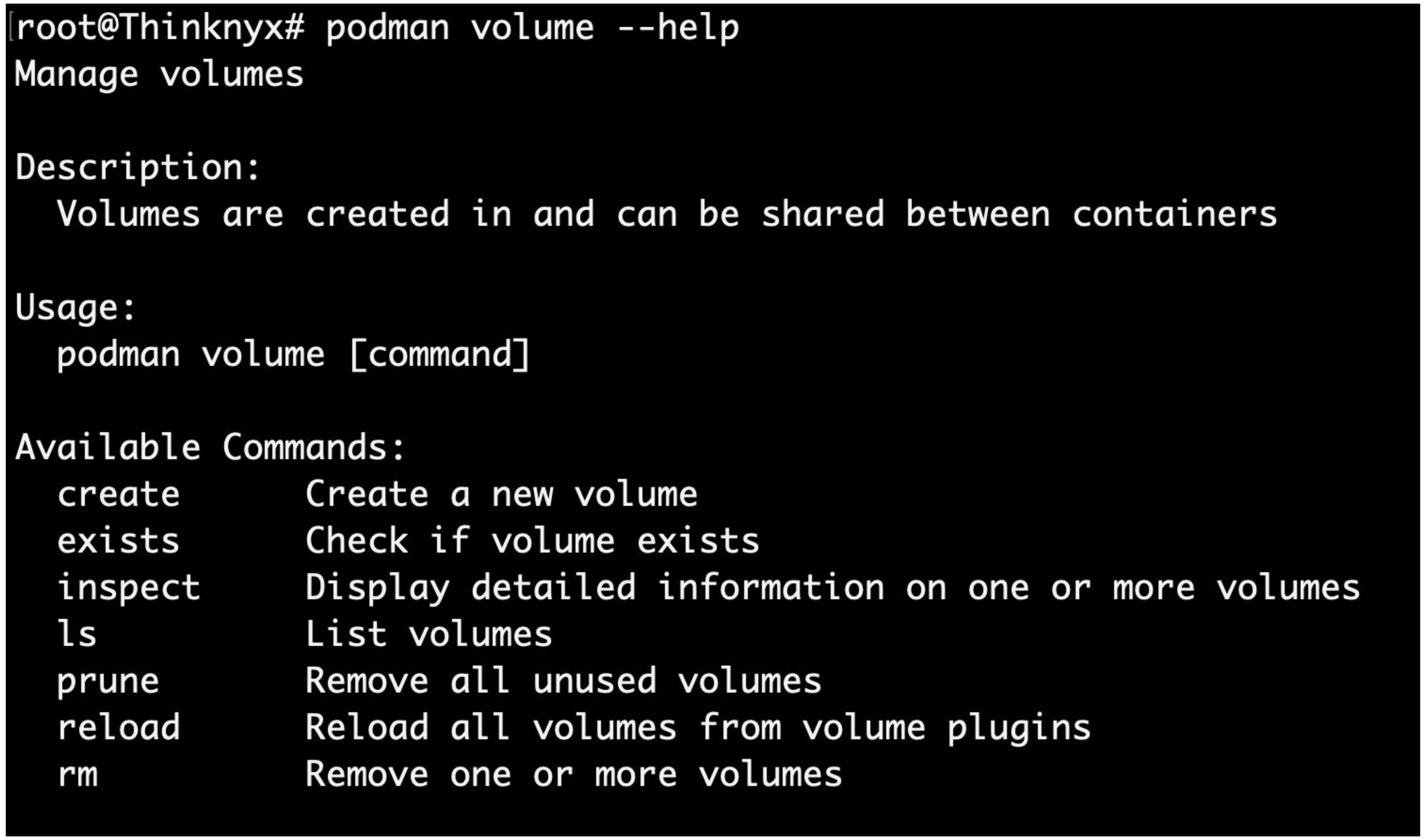
❑ Anonymous Volumes

Do not have a name
Created by Docker automatically
Randomly generated unique name and ID
Created when container is created
Manually mount to share across multiple containers

```
podman run -itd --name=thinknyxcon -p 8081:80 --mount
type=volume,target=/usr/local/apache2/htdocs httpd:latest
```

Managing volumes with Podman CLI

- *podman volume <sub-command> [options]*
- *podman volume --help*



```
root@Thinknyx# podman volume --help
Manage volumes

Description:
  Volumes are created in and can be shared between containers

Usage:
  podman volume [command]

Available Commands:
  create      Create a new volume
  exists      Check if volume exists
  inspect     Display detailed information on one or more volumes
  ls          List volumes
  prune       Remove all unused volumes
  reload      Reload all volumes from volume plugins
  rm          Remove one or more volumes
```

Managing volumes with Podman CLI

Description	Command	Alternative command
List all volumes	podman volume ls	podman volume list
Create a volume	podman volume create <volume_name>	--
View detailed information about a volume	podman volume inspect <volume_name>	--
Mount a named volume	podman container run --mount type=volume,src=<volume_name>,target=<container_mount_path> <image_name>	podman container run -v <volume_name>:<container_mount_path> <image_name>
Mount a Bind Mount volume	podman container run --mount type=bind,src=<directory_on_host>, target=<container_ mount_path> <image_name>	podman container run -v <directory_on_host>:<container_mount_path> <image_name>
Mount an anonymous volume	podman container run --mount type=volume,target=<container_mount_path> <image_name>	podman container run -v <container_mount_path> <image_name>
Remove a volume	podman volume rm <volume_name>	podman volume remove <volume_name>

Demonstration | Managing volumes with Podman CLI

Demonstration | Persisting data with Podman using Named volumes

Demonstration | Persisting data with Podman using Bind Mounts

Summary

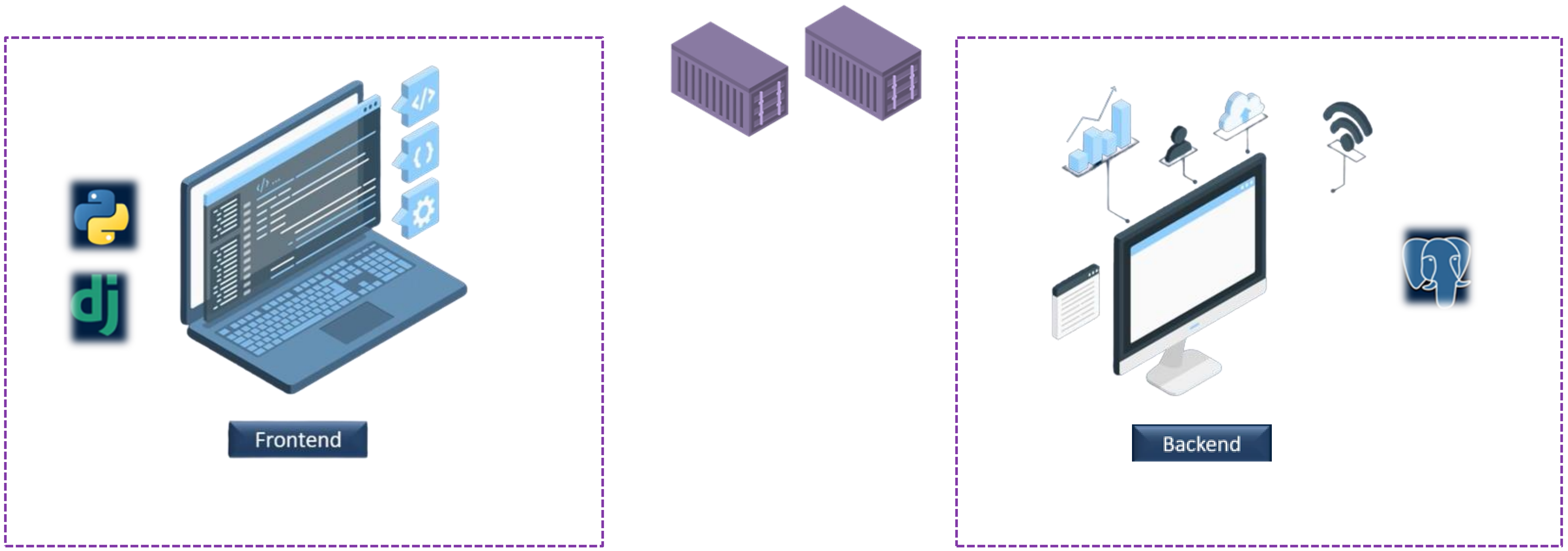
- Explored Storage available options in Podman
- Podman volumes Operations and data persistence in containers
- Demonstration

Podman | **Containerizing Applications**

Containerizing Applications

- Containerizing a Python application
  - Writing a Containerfile
  - Pushing the image to Docker Hub
  - Finally pulling that image to create and run a container
- Hands-on demonstrations

Demonstration | Creating Containerfile for our application


Frontend
Backend

Demonstration | Multi-stage Builds

Demonstration | Publishing Image to a Registry

Demonstration | Real time application deployment

Summary

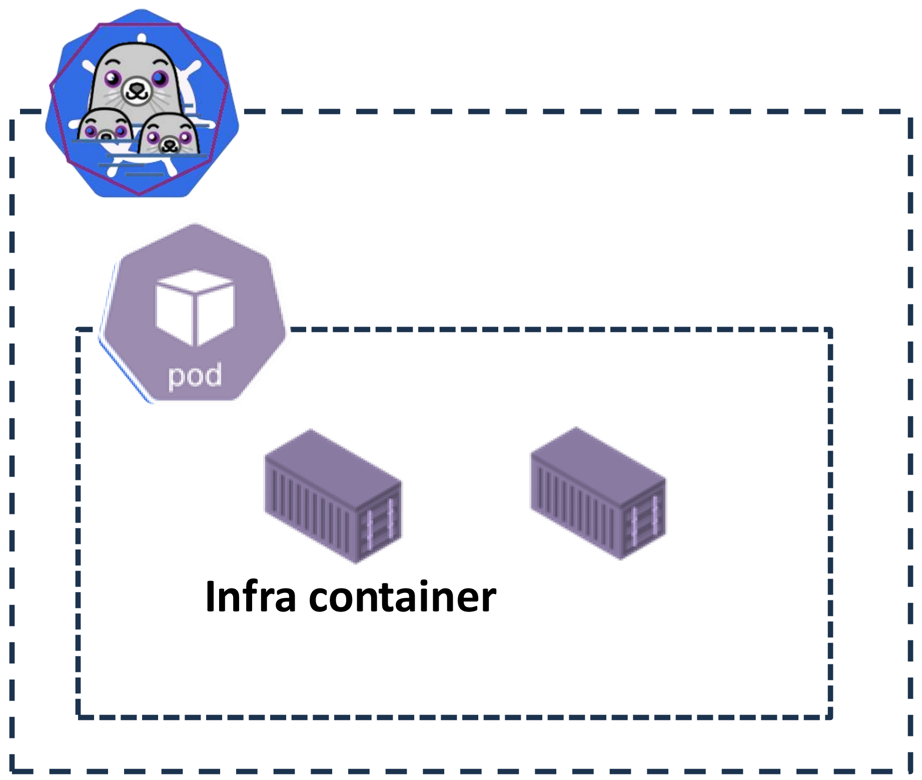
- Understood Build (image creation), Ship (publish) and Run (creating containers)
- Containerized Python based Application

Podman | **Pods in Podman**

Pods in Podman

- Introduction to Pods in Podman
- Pods Management in Podman

Pods in Podman



- Smallest installable units of computing
- Same network, uts, ipc namespaces

- ✓ Namespaces
- ✓ Port bindings
- ✓ Cgroup values

➢ **Infra container behaves like a paused container in Kubernetes**

Managing pods with Podman CLI

Description	Command	Alternative command
Create a pod	podman pod create	--
List a pod	podman pod ps OR podman pod ls	--
Create a container inside a pod	podman run -d --name <container_name> --pod <pod_name> <image_name>:tag	--
List all containers associated with a pod	podman ps -a --pod | grep -i <pod_name>	--
Inspect a pod	podman pod inspect <pod_name>	--
Stop a Pod	podman pod stop <pod_name>	--
Start a Pod	podman pod start <pod_name>	--
Remove a pod	podman pod rm <pod_name>	--

Demonstration | Managing pods with Podman CLI

Demonstration | Accessing Containers within a Pod in Podman

Summary

- ❖ Pods in Podman
- ❖ Why Infra containers are crucial for Pods