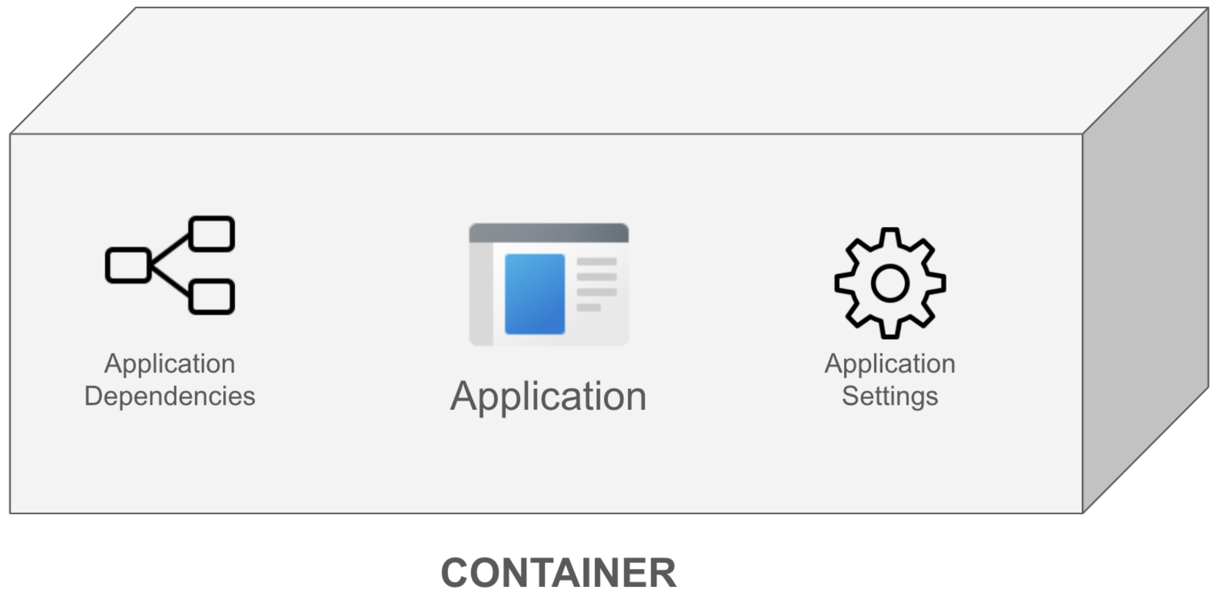
Containerization with Docker

CONTAINERIZATION AND VIRTUALIZATION CONCEPTS

Julia Ostheimer
Freelance AI Consultant

Recap: Definition of a container

- Containers
  - Isolated environment
  - Includes application and all dependencies



[1] Icons by icons8.com

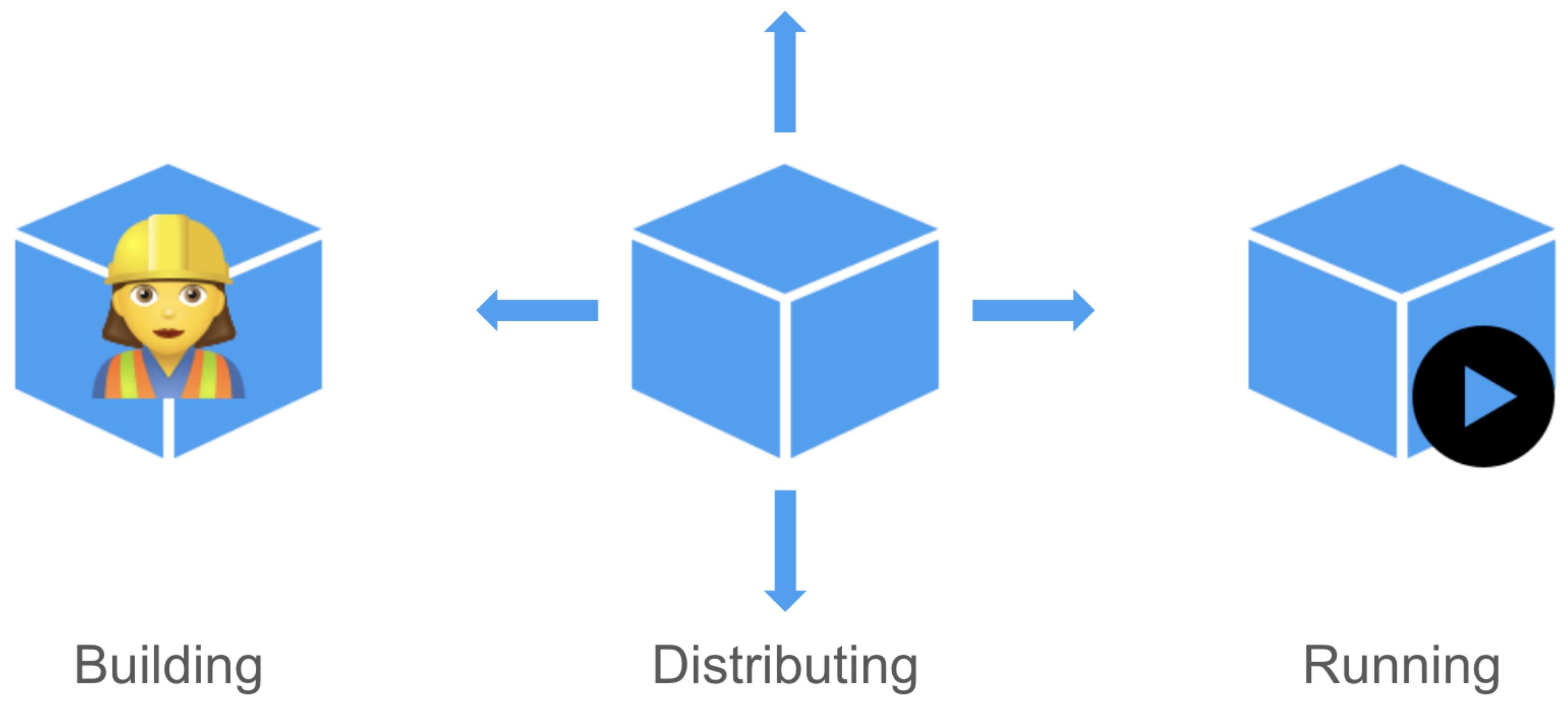
Introducing Docker

- The go-to containerization tool
- Open-source & large user base
- One of the most used and popular tools!

[1] Stack Overflow Developer Survey from 2023 [2] Logo by Docker Inc. from 2024

Introducing Docker

- Managing the lifecycle of containers

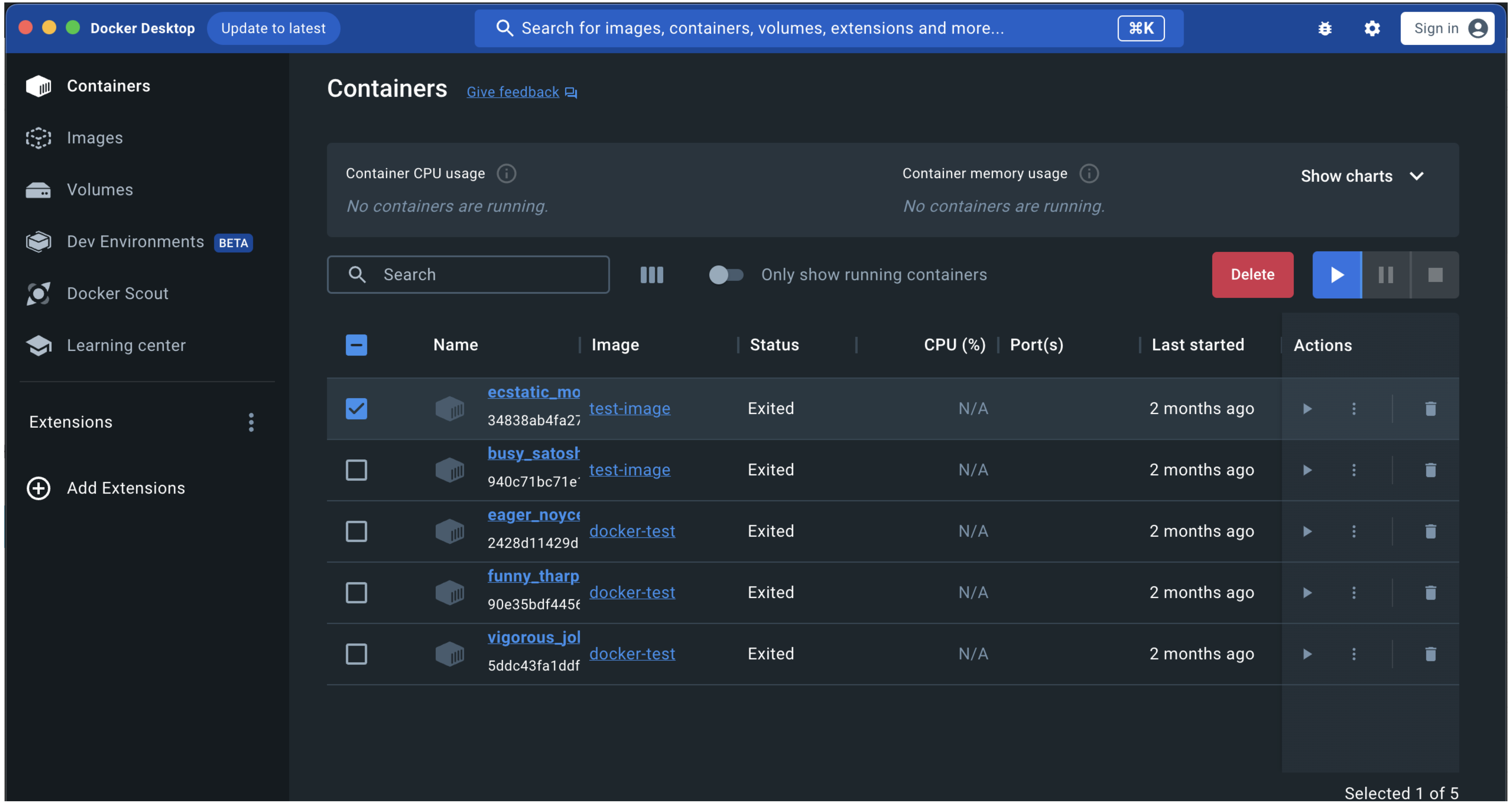


[1] Icons by icons8.com

Overview of Docker components

- Most important Docker components:
  - **Docker Desktop**
  - **Docker Engine**
    - Docker Client
    - Docker Daemon
  - **Docker Objects**
    - Docker Images
    - Docker Containers
  - **Docker Registries**

Installing Docker via Docker Desktop



Containers

Container CPU usage
No containers are running.

Container memory usage
No containers are running.

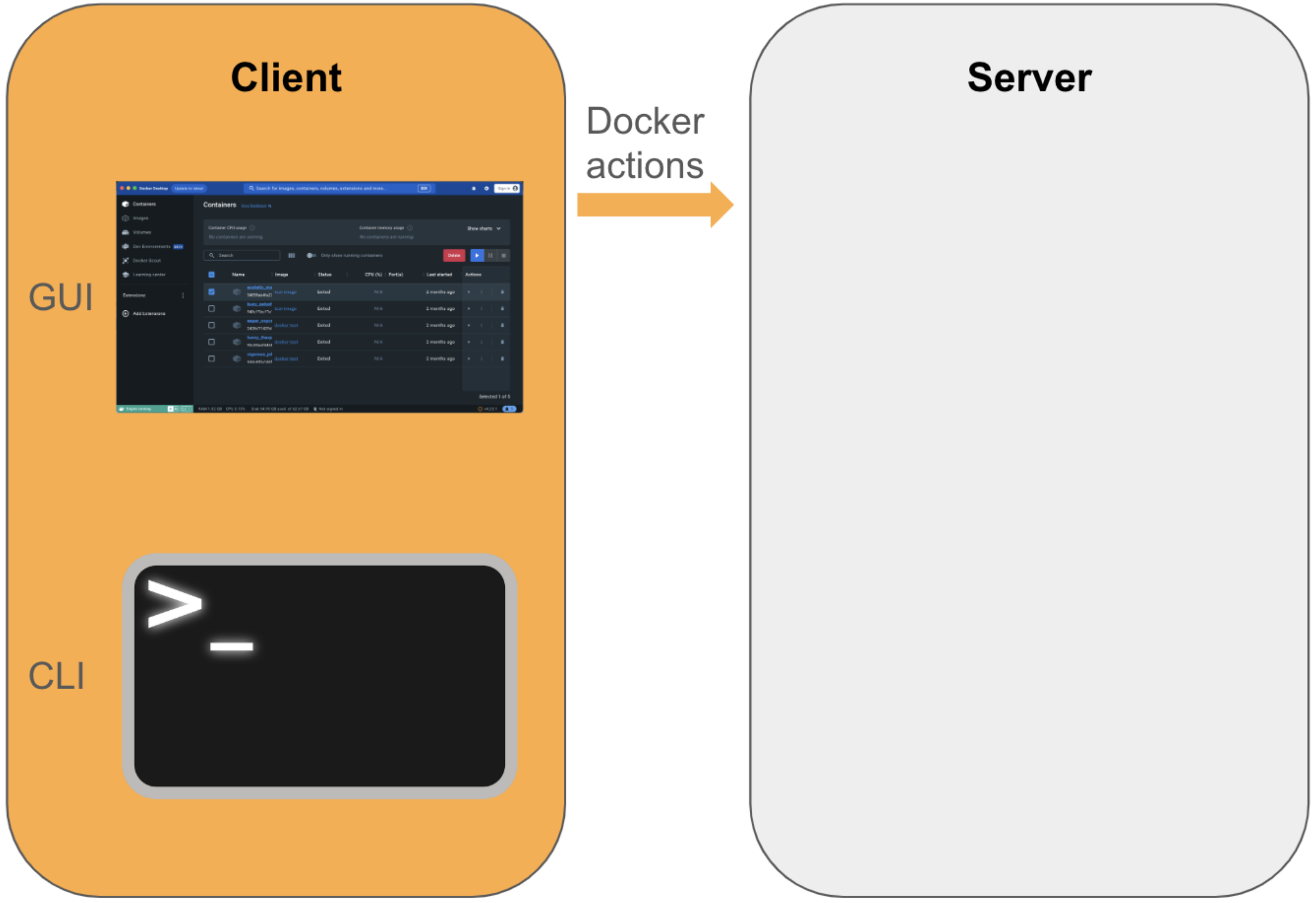
	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
✓	ecstatic_mo 34838ab4fa27	test-image	Exited	N/A		2 months ago	
	busy_satosh 940c71bc71e	test-image	Exited	N/A		2 months ago	
	eager_noyce 2428d11429d	docker-test	Exited	N/A		2 months ago	
	funny_tharp 90e35bdf4456	docker-test	Exited	N/A		2 months ago	
	vigorous_jol 5ddc43fa1ddf	docker-test	Exited	N/A		2 months ago	

Selected 1 of 5

[1] Screenshot from Docker Desktop Mac application

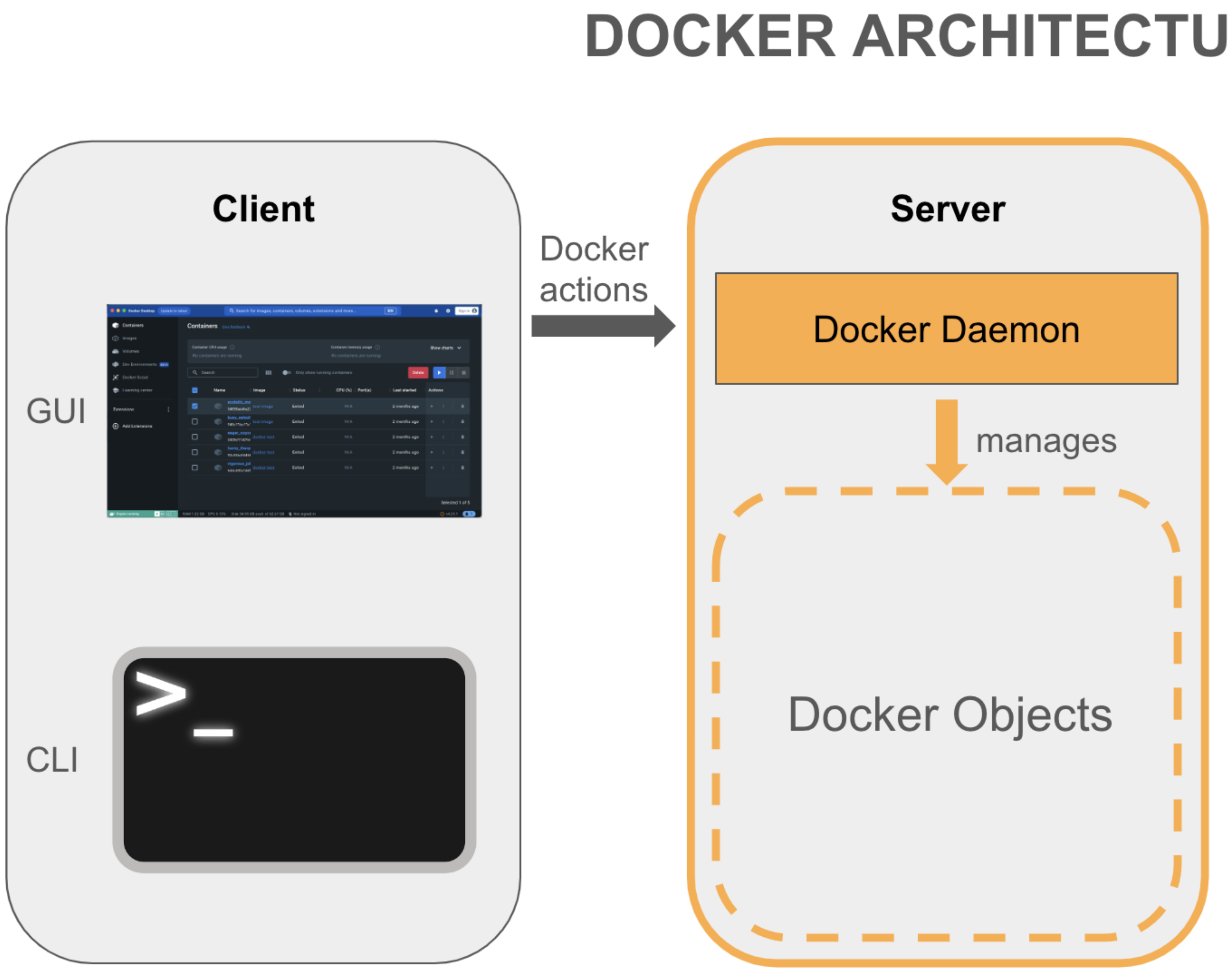
Client-server architecture of Docker Engine

DOCKER ARCHITECTURE



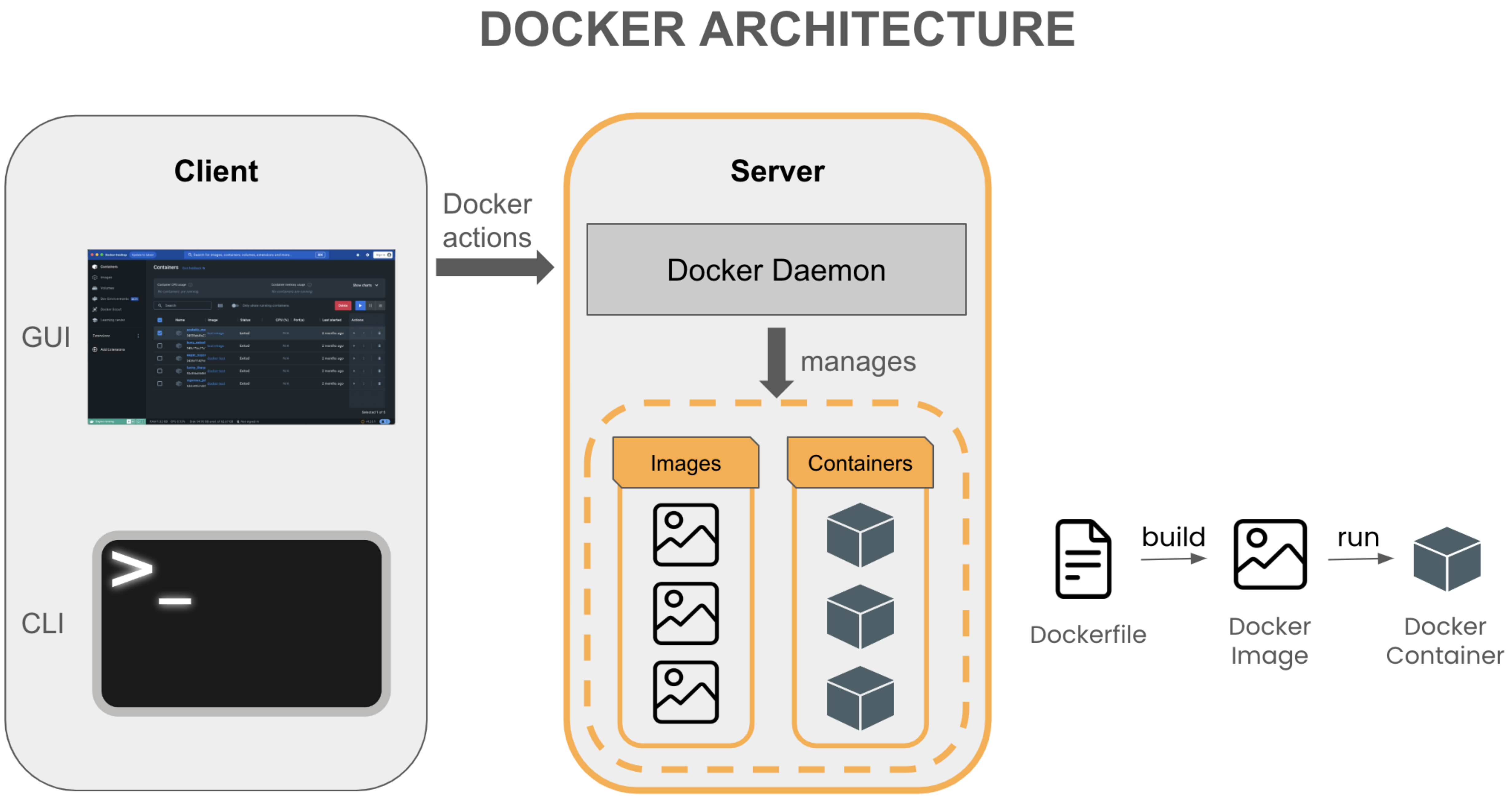
[1] Icons by icons8.com

Client-server architecture of Docker Engine



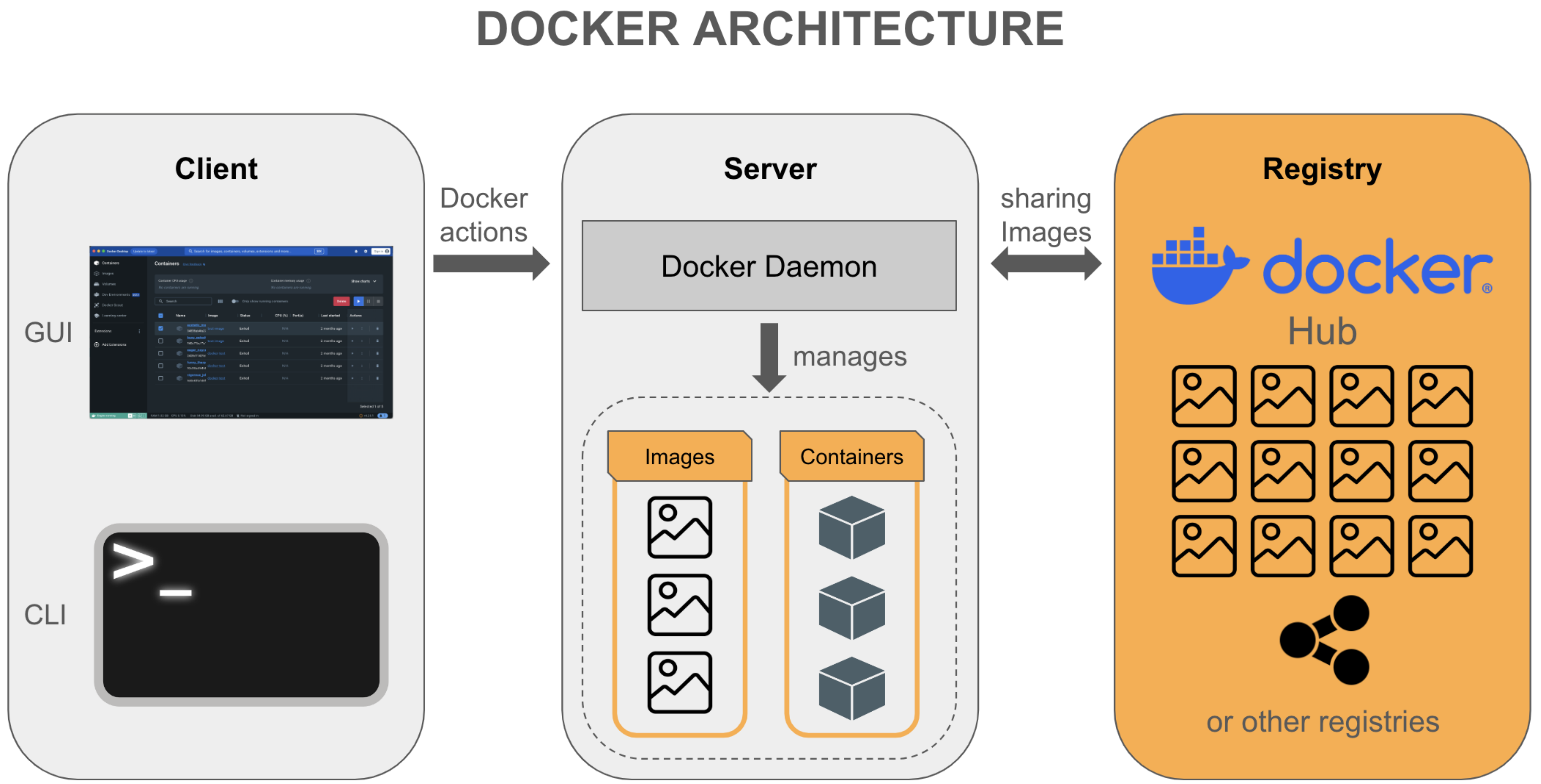
[1] Icons by icons8.com

Overview of Docker objects



[1] Icons by icons8.com

Sharing containers via registries



[1] Icons by icons8.com

Let's practice!

CONTAINERIZATION AND VIRTUALIZATION CONCEPTS

Container orchestration

CONTAINERIZATION AND VIRTUALIZATION CONCEPTS

Julia Ostheimer
Freelance AI Consultant

Definition of container orchestration

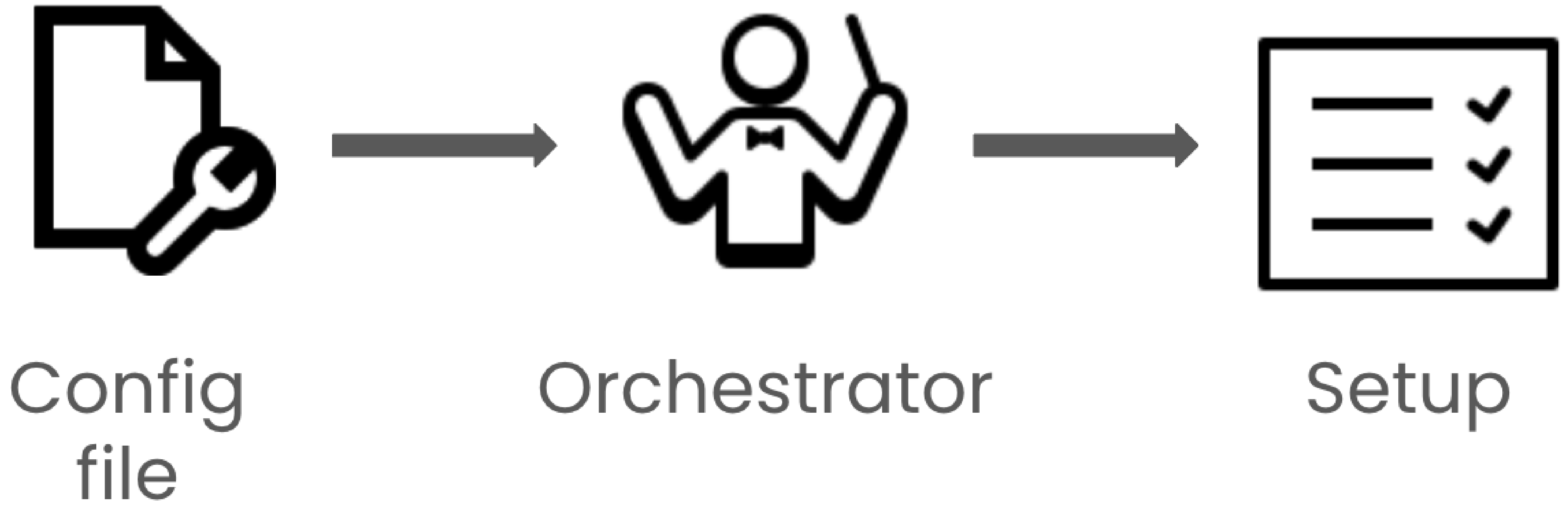
- **Orchestration:**
  - The automated management of multiple components
- **Orchestrator:**
  - The tool used for orchestration
- **Container Orchestration:**
  - Orchestration of containers

Purpose of container orchestration

- Simplifies management of many containers
- Ensuring that multiple containers interact **effectively and efficiently**

Declarative programming in container orchestration

- Declarative programming:
  - Defining the desired output instead of describing the steps to reach it



[1] Icons by Icons8.com

Benefits of container orchestration

- Easy **scaling** of containers
  - Horizontal scaling: Adding/Removing containers
  - Vertical scaling: In-/Decreasing computing resources of specific containers
- **Automation** of operations
  - Time savings
  - Improved developer productivity
  - Cost savings
- Better **performance** of application

Applications of container orchestration

- Microservices architecture
- Application scaling
- Automation of pipelines

Container orchestration tools

Swarm mode

HashiCorp Nomad

OpenShift

kubernetes

[1] Logos by Docker Inc., Red Hat Inc., HashiCorp Inc., and The Linux Foundation

Let's practice!

CONTAINERIZATION AND VIRTUALIZATION CONCEPTS

Container orchestration with Kubernetes

CONTAINERIZATION AND VIRTUALIZATION CONCEPTS

Julia Ostheimer
Freelance AI Consultant

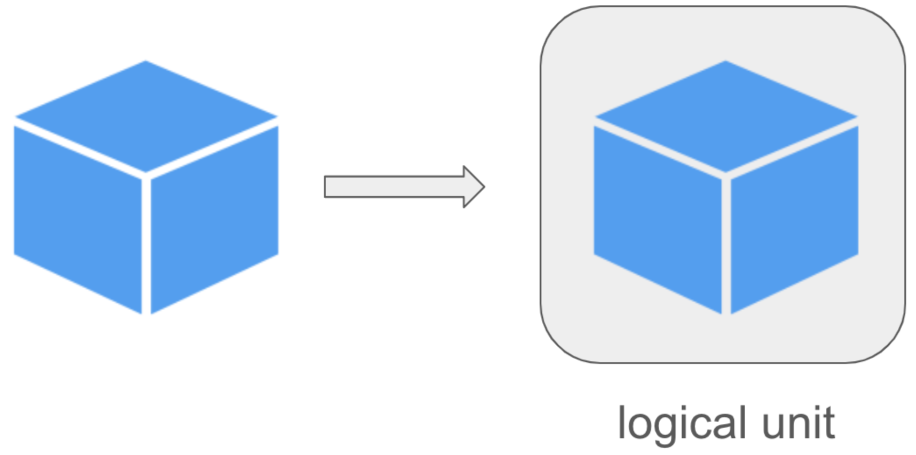
Introducing Kubernetes

- Abbreviation: K8s
- Developed by Google, open-sourced in 2014
- 96% of organizations use/evaluate using Kubernetes

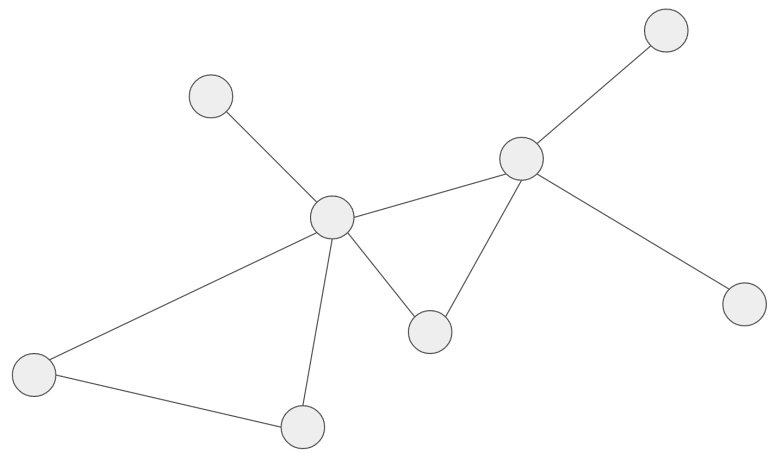
[1] Cloud Native Computing Foundation (CNCF) Annual Survey in 2022 [2] Logo by The Linux Foundation

Introducing Kubernetes

- Grouping containers into logical units



- Distributed system

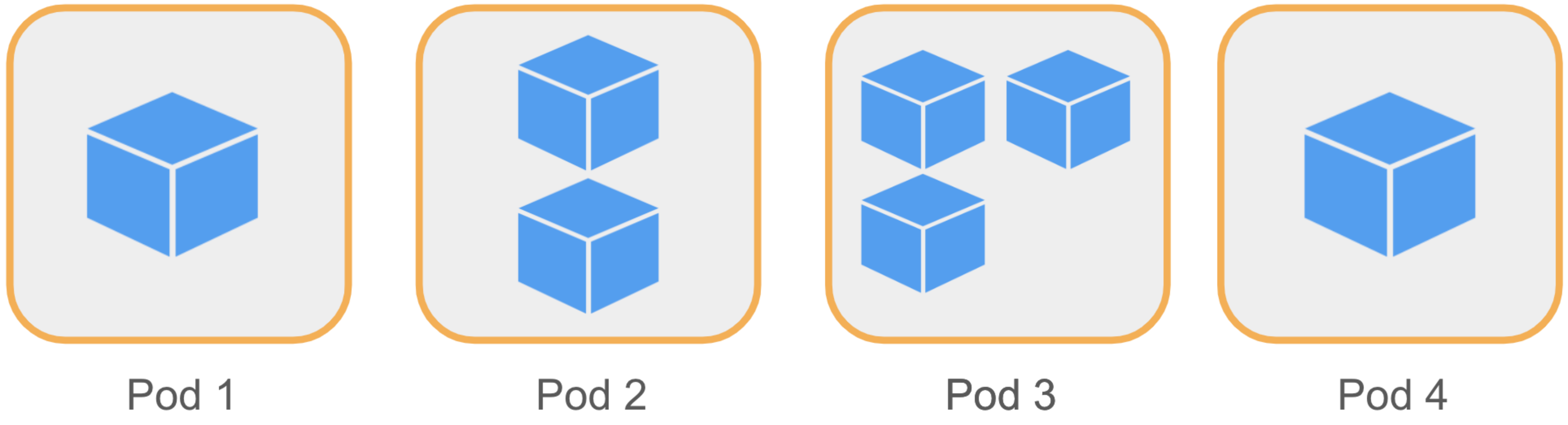


[1] Icons by Icons8.com

Overview of Kubernetes components

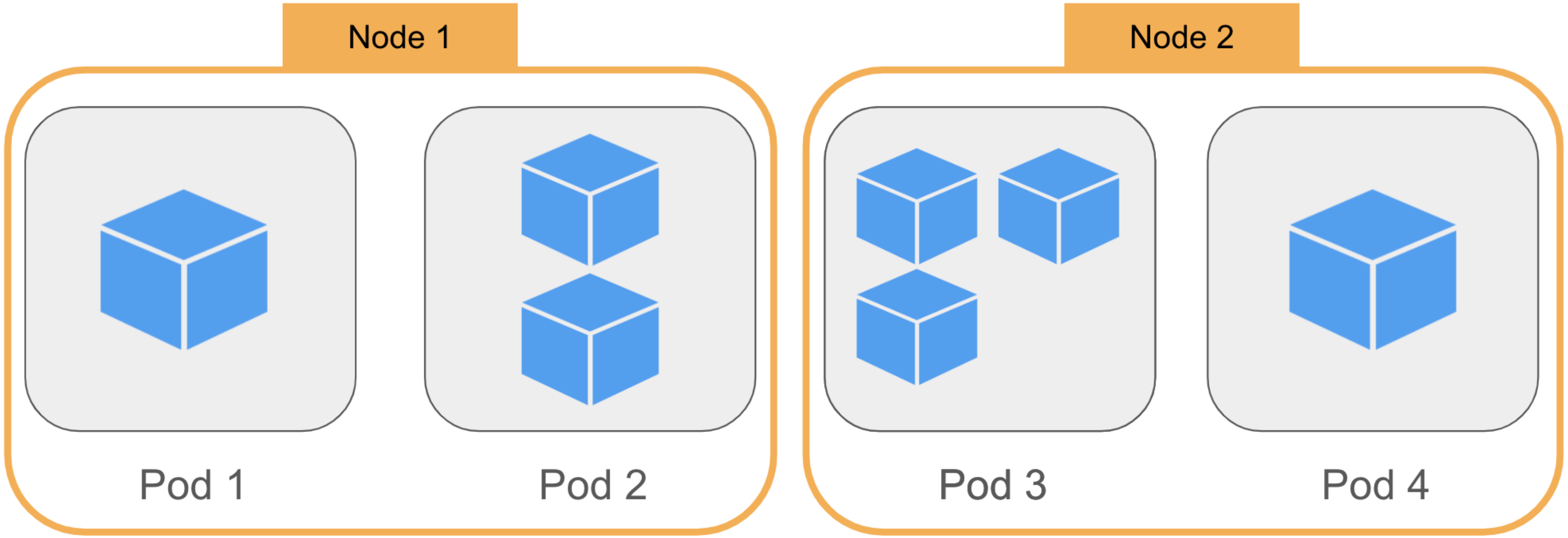
- Most important Kubernetes components:
  - Pods
  - Nodes
  - Control Plane
  - Cluster

Pods as smallest deployable unit



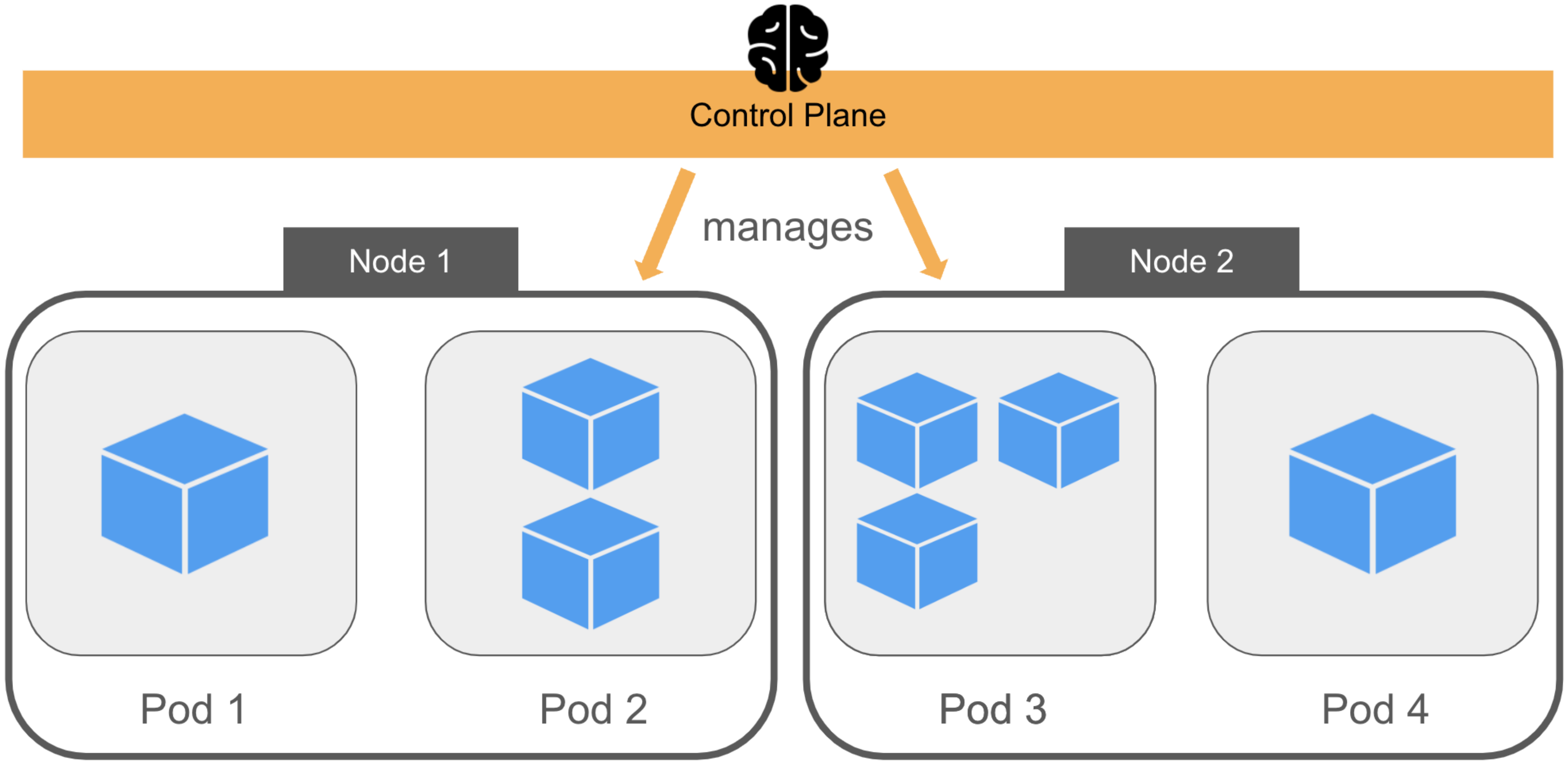
[1] Icons by Icons8.com

Nodes as smallest hardware unit



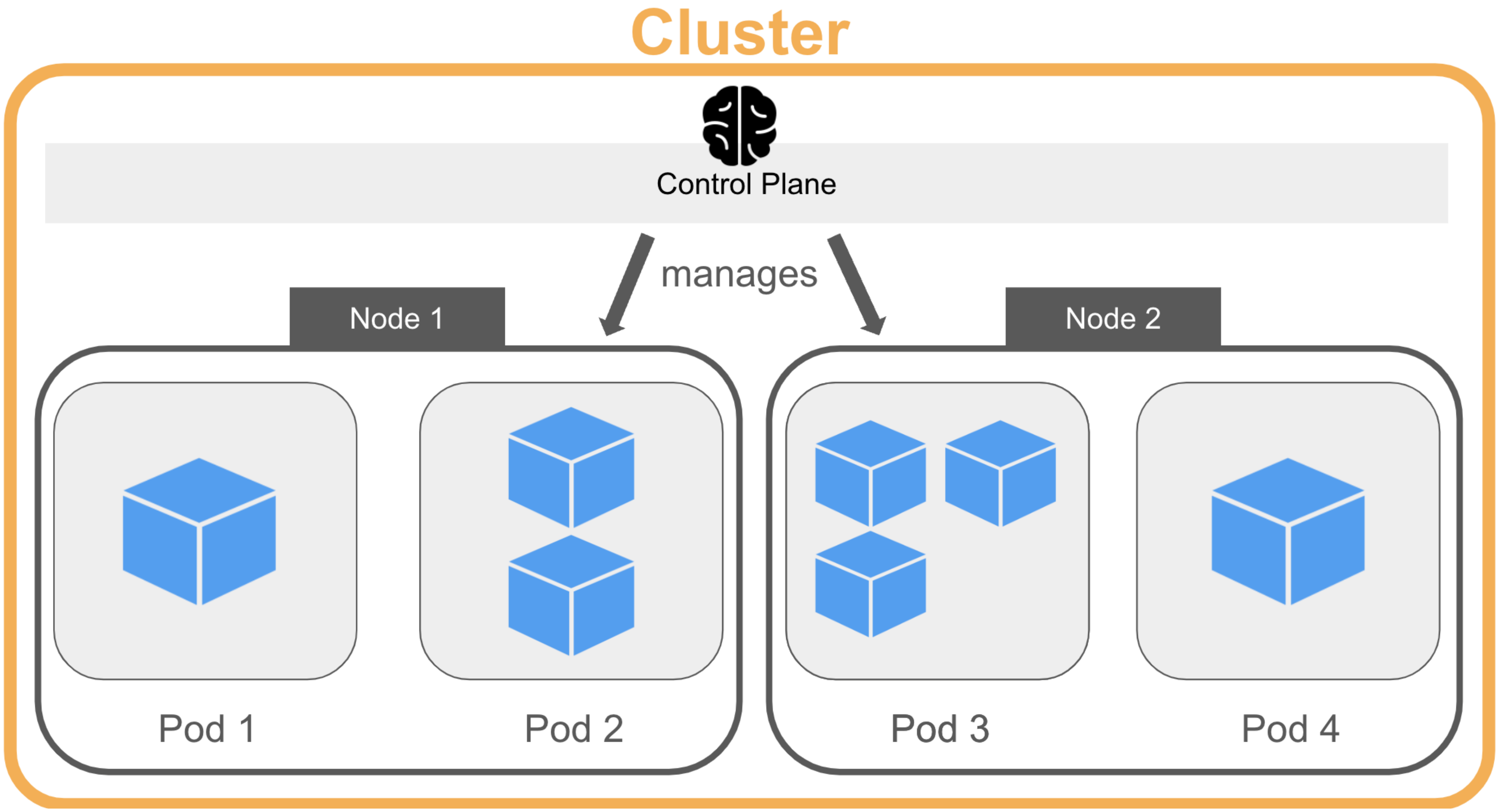
[1] Icons by Icons8.com

Node management via control plane



[1] Icons by Icons8.com

Grouping nodes in a cluster



[1] Icons by Icons8.com

Docker and Kubernetes

- Docker: Dealing with **one or few containers**

- Kubernetes: Dealing with **many containers**

Let's practice!

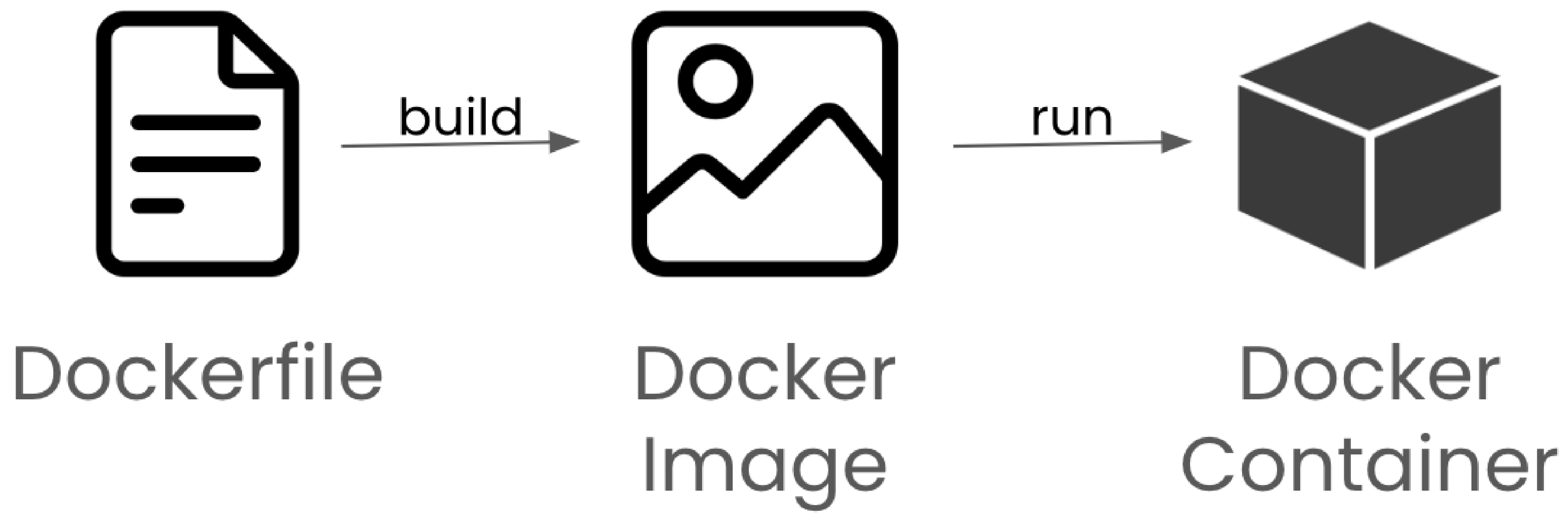
CONTAINERIZATION AND VIRTUALIZATION CONCEPTS

Reading Dockerfiles and running containers

CONTAINERIZATION AND VIRTUALIZATION CONCEPTS

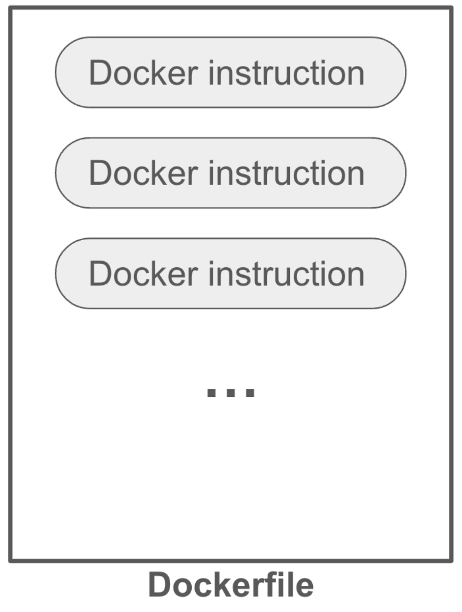
Julia Ostheimer
Freelance AI Consultant

Recap of Docker terms



Docker instructions vs. Docker commands

- Docker instructions detail how to build a Docker image



- Docker commands: Commands via Command Line Interface (CLI)

```
>docker command
```

Format of a Dockerfile

comment

```
# Define the image on which to build
FROM python:3.10

...
```

Format of a Dockerfile

```
# Define the image on which to build
FROM python:3.10

...
```

Docker instruction

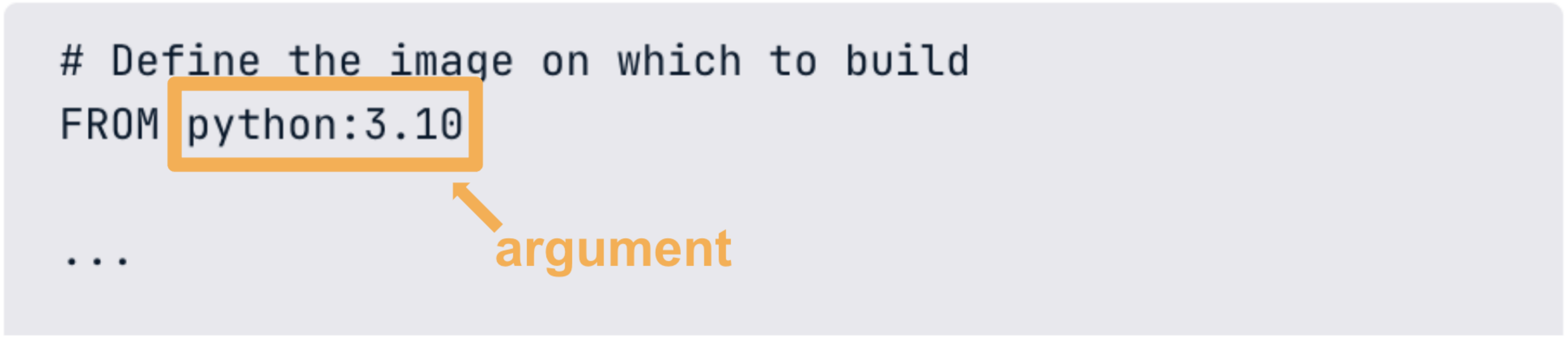
Format of a Dockerfile

```
# Define the image on which to build
FROM python:3.10

...
```

COMMAND

Format of a Dockerfile



```
# Define the image on which to build
FROM python:3.10

...
```

Sequential order in Dockerfiles

- Execution in sequential order
- Start of a Dockerfile:
  - Metadata
  - Comments
  - Arguments
  - `FROM` instruction

Overview of Docker instructions

- Important Docker instructions
  - `FROM`
  - `COPY`
  - `RUN`
  - `ENTRYPOINT`

FROM instruction

- Specifies an existing Docker image
- Defines the image we are building on
  - "Starting point"

Syntax:

```
FROM <name_of_image>
```

Example:

```
# Define the image on which to build
FROM python:3.10
```

COPY instruction

- Copies files or directories
  - From source (<source>) to destination (<destination>)
  - Files that are needed in following Docker instructions

Syntax:

```
COPY <source> <destination>
```

Example:

```
# Copy files/folders to the main folder of the container
COPY . .
```

RUN instruction

- Runs a command within a container
  - Can be any command that could be run in a CLI

Syntax:

```
RUN <command>
```

Example:

```
# Install the application's dependencies
RUN pip install -r requirements.txt
```

ENTRYPOINT instruction

- Defines the container's default behavior
  - Specifies command to run at initiation
  - The primary purpose of the container

Syntax:

```
ENTRYPOINT ["command", "argument"]
```

Example:

```
# Run the script when the container starts
ENTRYPOINT ["python", "hello_world.py"]
```

Assembling instructions to Dockerfile

```
# Define the image on which to build
FROM python:3.10
```

Assembling instructions to Dockerfile

```
# Define the image on which to build
FROM python:3.10

# Copy files/folders to the main folder of the container
COPY . .
```

Assembling instructions to Dockerfile

```
# Define the image on which to build
FROM python:3.10

# Copy files/folders to the main folder of the container
COPY . .

# Install the application's dependencies
RUN pip install -r requirements.txt
```

Assembling instructions to Dockerfile

```
# Define the image on which to build
FROM python:3.10

# Copy files/folders to the main folder of the container
COPY . .

# Install the application's dependencies
RUN pip install -r requirements.txt

# Run the script when the container starts
ENTRYPOINT ["python", "hello_world.py"]
```

Docker build command

- Builds Docker image from a Dockerfile
    - Dockerfile needs to be located in build's context (<context>)
    - Executed as command with Docker client via CLI

Syntax:

```
docker build <context>
```

Docker run command

- Creates and runs Docker container from Docker image
  - Docker image needs to be specified as argument (<name_of_image>)
  - Executed as command with Docker client via CLI

Syntax:

```
docker run <name_of_image>
```

Let's practice!

CONTAINERIZATION AND VIRTUALIZATION CONCEPTS

Wrap-up

CONTAINERIZATION AND VIRTUALIZATION CONCEPTS

Julia Ostheimer
Freelance AI Consultant

Recap of course goals

- Chapter 1
  - Define virtualization
  - Define containerization
  - Comparing containerization and virtualization

- Chapter 2
  - Explain containerization with Docker
  - Define container orchestration
  - Explain container orchestration with Kubernetes
  - Hands-on with Docker

Recap: Virtualization vs. containerization

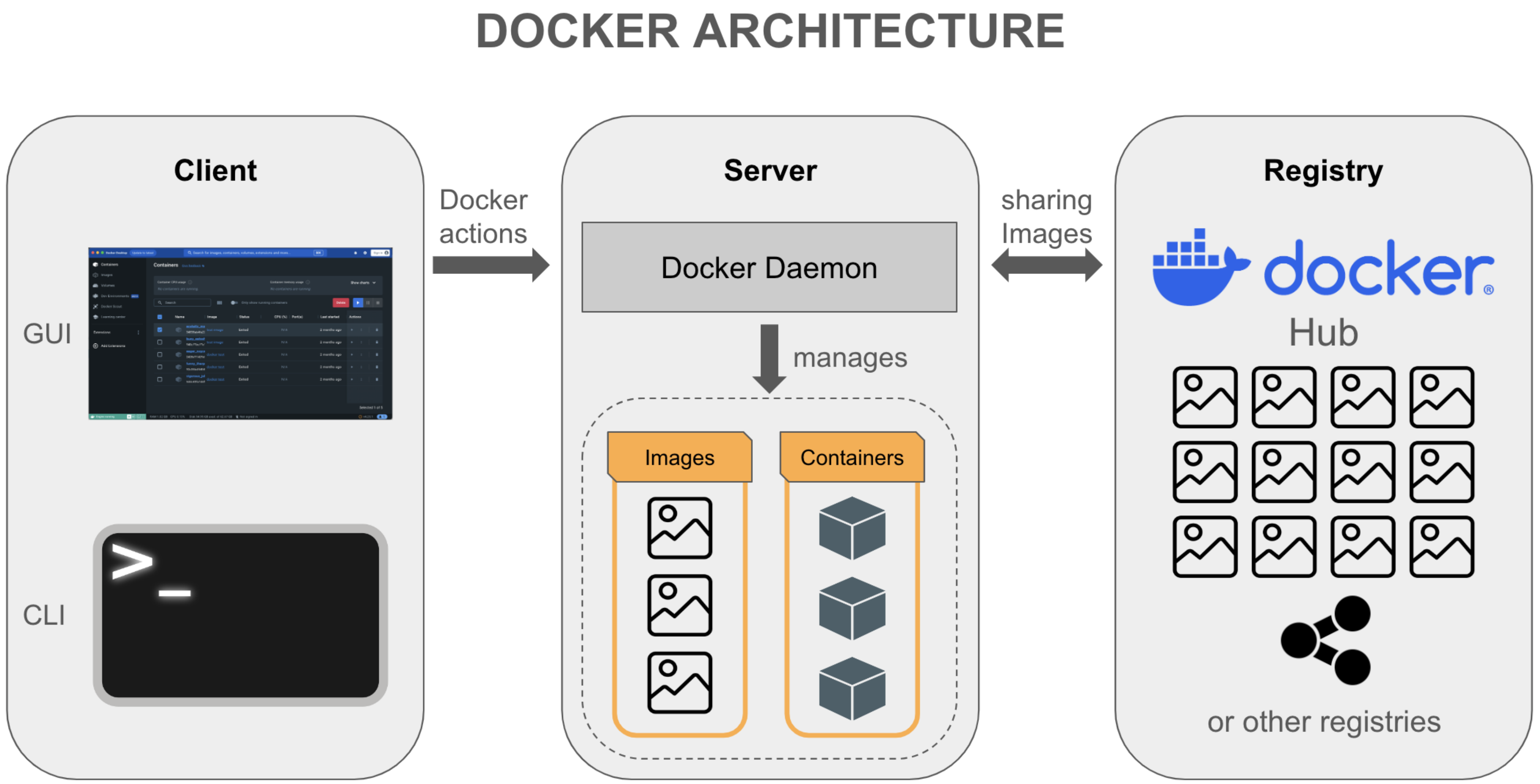
Virtualization

- Creates a virtual version of a computing resource
- Full virtualization
- VM: Simulated computer system inside another computer

Containerization

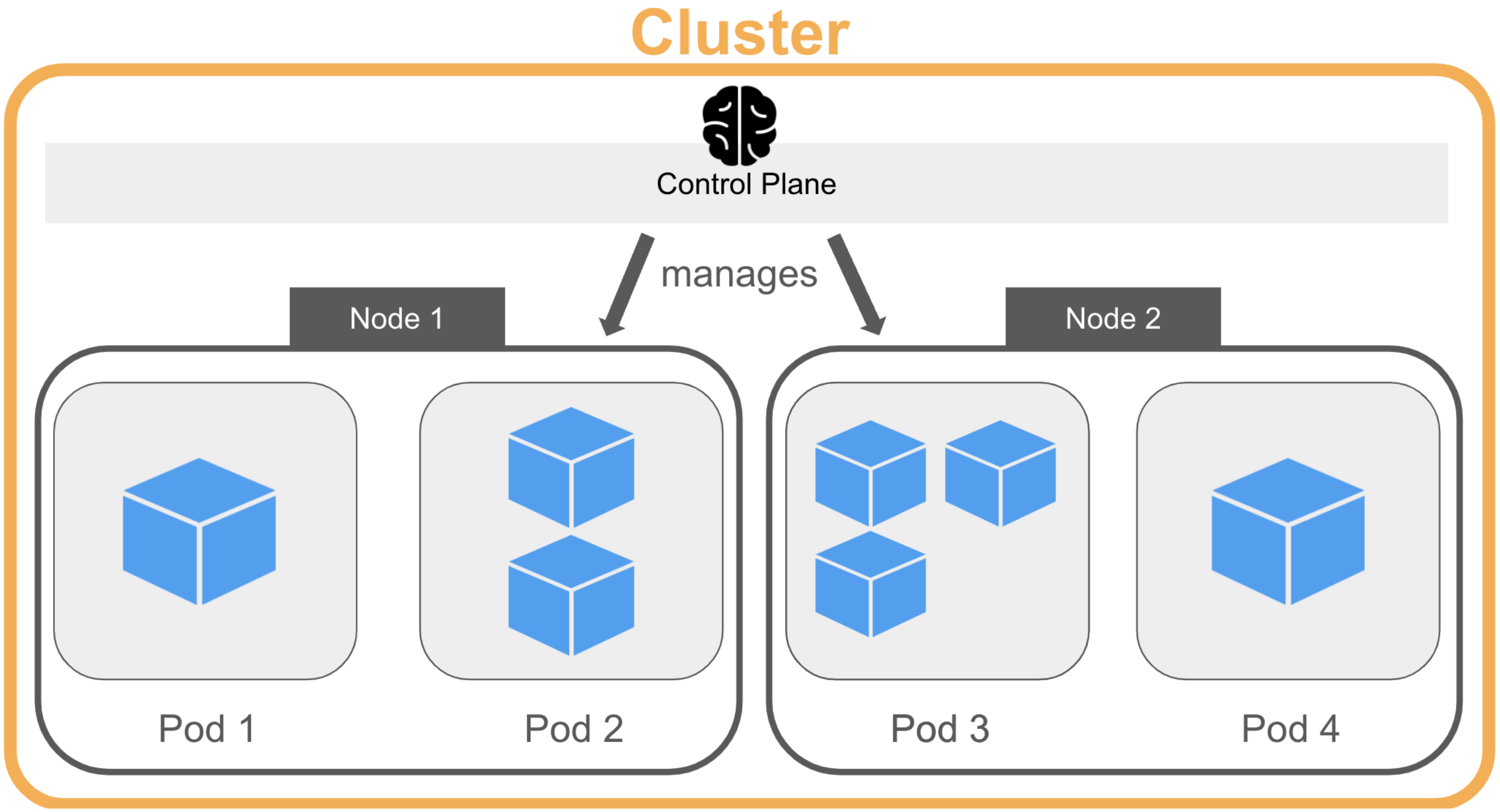
- Packages application and dependencies into isolated environment
- OS-level virtualization
- Container: Isolated application environment

Docker architecture



[1] Icons by Icons8.com

Kubernetes architecture



[1] Icons by Icons8.com

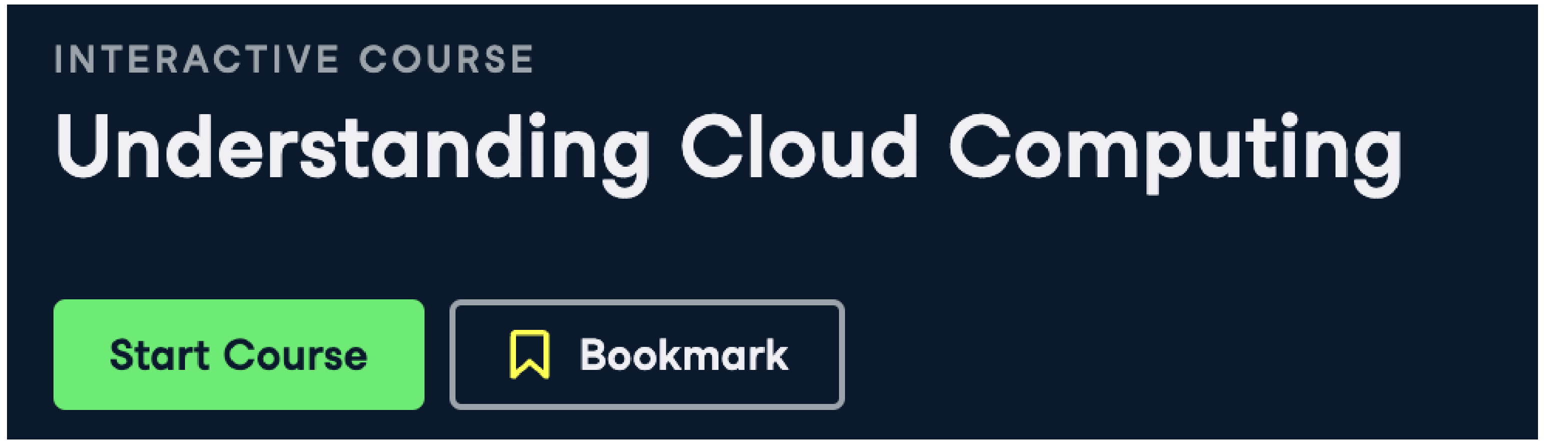
Docker instructions and commands

Docker instruction	Description
`FROM`	Defines the image to build on.
`COPY`	Copies files or directories into the container.
`RUN`	Runs a command inside the container.
`ENTRYPOINT`	Defines the default behavior of the container.

Docker command	Description
`docker build <context>`	Builds a Docker image based on Dockerfile.
`docker run <name_of_image>`	Runs a Docker container based on Docker image.

Hungry for more?

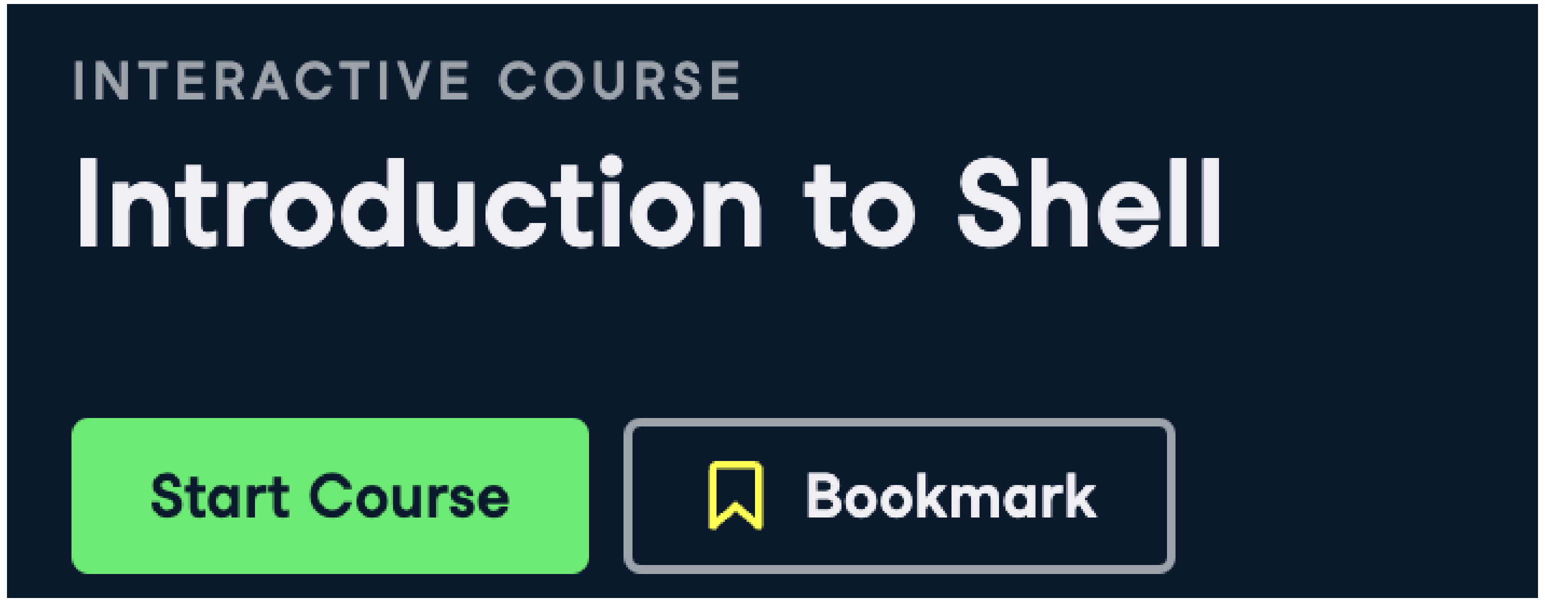
- Understanding computing in the cloud



INTERACTIVE COURSE

Understanding Cloud Computing

- Dealing with CLI



INTERACTIVE COURSE

Introduction to Shell

Hungry for more?

- **Continue learning about Docker & Kubernetes!**
  1. `DONE` Introduction to Containerization and Virtualization
  2. `UPCOMING` Introduction to Docker
  3. Intermediate Docker
  4. Introduction to Kubernetes

Congratulations!

CONTAINERIZATION AND VIRTUALIZATION CONCEPTS