Containerized Apps

With Docker and Podman

What are Containers?

Containers are lightweight **packages of your application** code together **with dependencies** such as specific versions of programming language runtimes and libraries required to run your software services.

Source: What are containers? | Google Cloud

Linux Containers

We are talking about Linux containers.

They are a set of one or more processes which are **isolated** from the rest of the system.

All the files necessary to run them are provided from a distinct image, thus they are **portable and consistent** as they move from development, to testing, and finally to production.

Linux Container LXC

The Linux Containers project (LXC) is an **open-source container platform that provides a set of tools**, templates, libraries, and language bindings. LXC has a simple command line interface that improves the user experience when starting containers.

LXC offers an operating-system level virtualization environment that is available to be installed on many Linux-based systems. Your Linux distribution may have it available through its package repository.

Why Container

- Separation of responsibility
  - developers focus on application logic and dependencies, while IT operations teams can focus on deployment and management

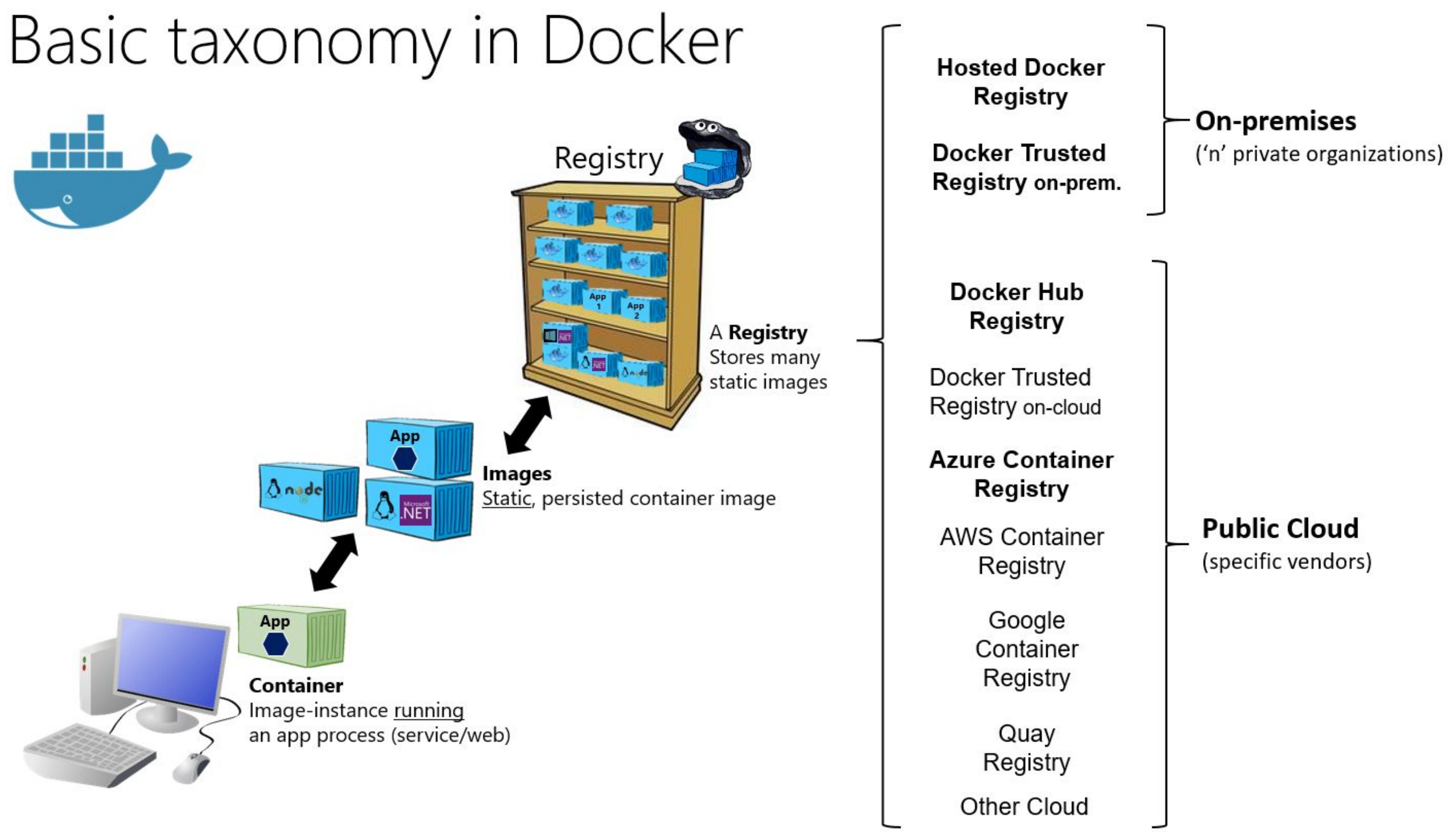
- Workload portability
  - can run virtually anywhere

- Application isolation
  - virtualize CPU, memory, storage, and network resources at the operating system level. Developers get a view of the OS logically isolated from other applications.

- Enables scalability
  - Easy to reproduce

Based on: What are containers? | Google Cloud

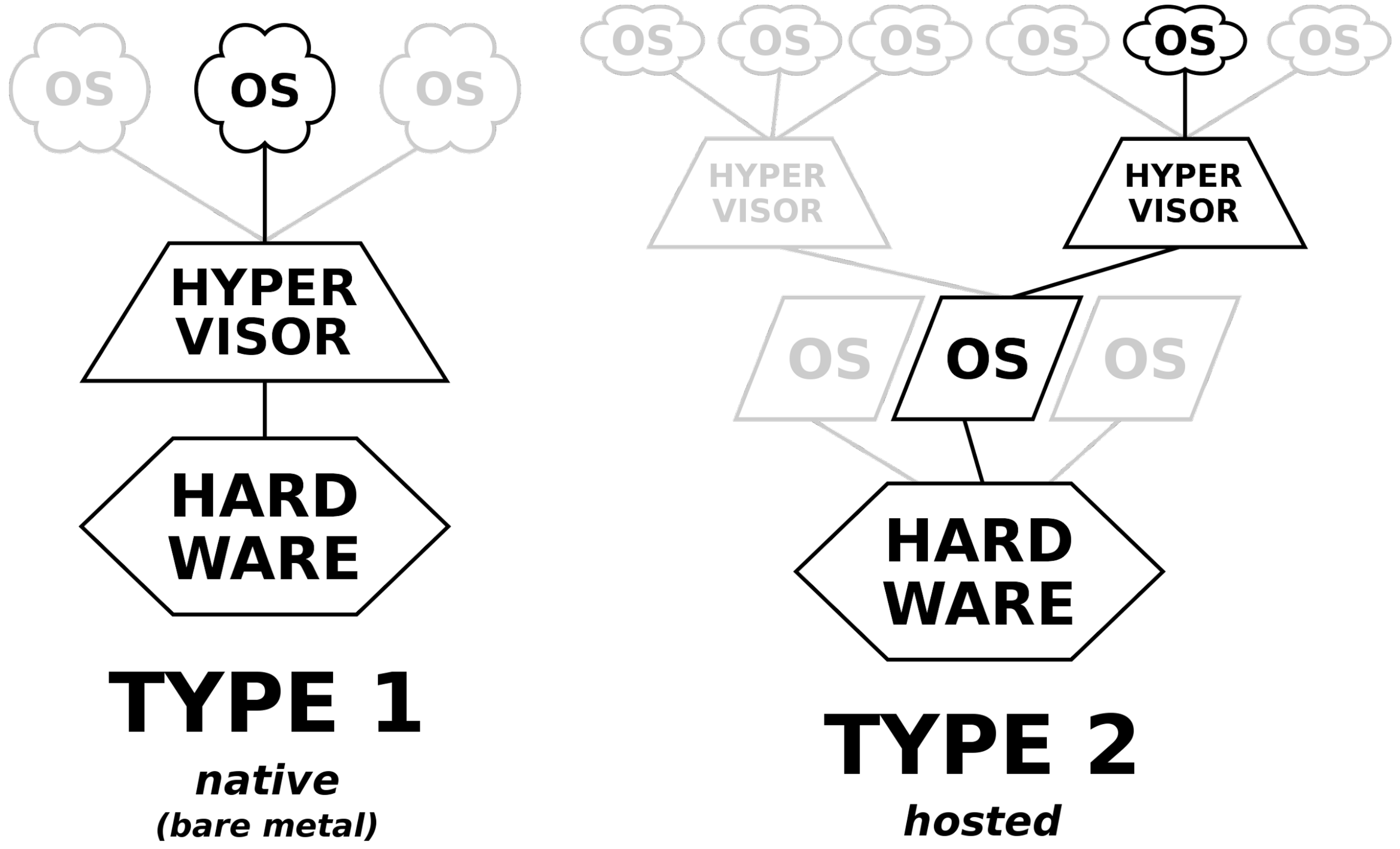
Container Registry


Basic taxonomy in Docker
Registry
A **Registry** Stores many static images
App 1
App 2
App
Images
Static, persisted container image
App
Container
Image-instance running an app process (service/web)
Hosted Docker Registry
Docker Trusted Registry on-prem.
On-premises
('n' private organizations)
Docker Hub Registry
Docker Trusted Registry on-cloud
Azure Container Registry
AWS Container Registry
Google Container Registry
Quay Registry
Other Cloud
Public Cloud
(specific vendors)

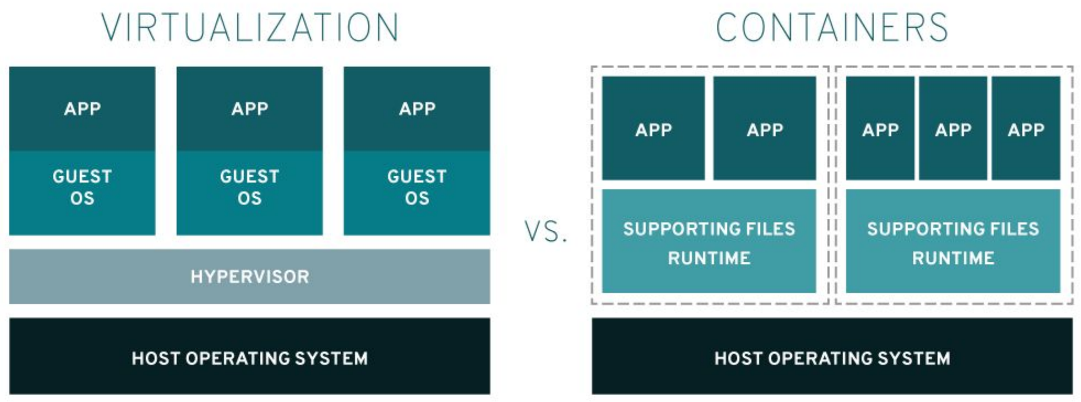
Virtualization

- Virtualization lets your operating systems run simultaneously on a single hardware system.
- It uses a hypervisor for partitioning the resource of a CPU among multiple operating systems or independent programs.

Hypervisors



Container vs VMs



Source: What's a Linux container? (redhat.com)

Containers allow you to package your application together with libraries and other dependencies, providing isolated environments for running your software services just like virtual machines. That is all with the similarities, since containers:

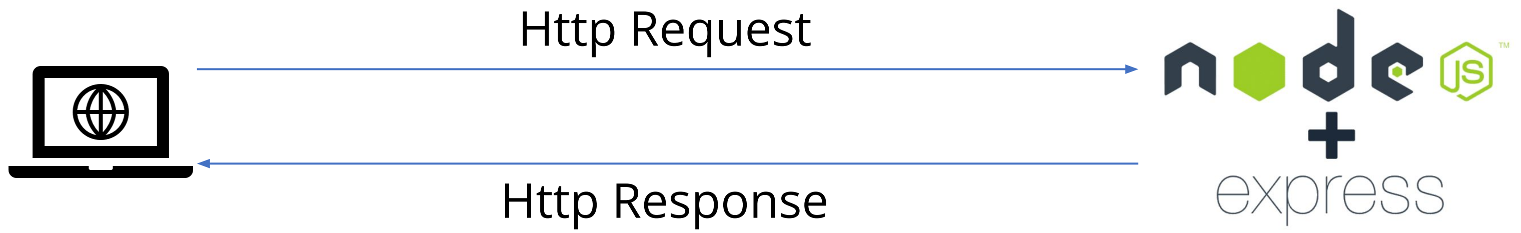
- are much more lightweight than VMs
- virtualize at the OS level while VMs virtualize at the hardware level
- share the OS kernel and use a fraction of the memory VMs require

Based on: What are containers? | Google Cloud

How to Setup a Container

- They are configured with a Containerfile (Dockerfile)
- The file will be executed from top to bottom
- If the image is not available, it will be downloaded from the registry. Mostly from Docker Hub

Example Pizza Service



Setup

1. Create Containerfile
2. Download image
3. Inside the image create app directory
4. Install app dependencies
5. Bundle app source inside the container image
6. Bind ports from inside the container to the outside
7. Optional, but advisable for NodeJS; create a .dockerignore file
8. Start the application

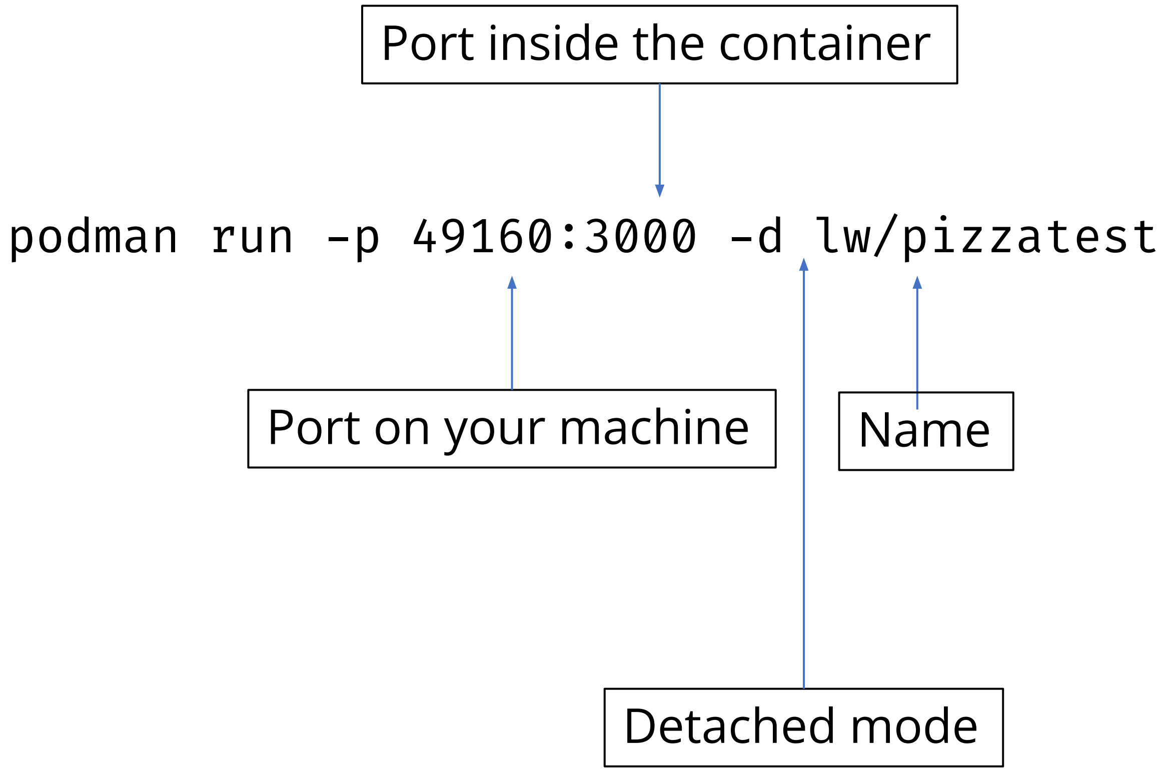
Containerfile

```
# Get the image from dockerhub
FROM node:18

# Create app directory
WORKDIR /usr/src/app

# A wildcard is used to ensure both package.json AND
package-lock.json are copied
COPY package*.json ./
# Install app dependencies
RUN npm install

# Bundle app source
COPY . .

EXPOSE 3000
CMD [ "node", "index.js" ]
```

Build the Container

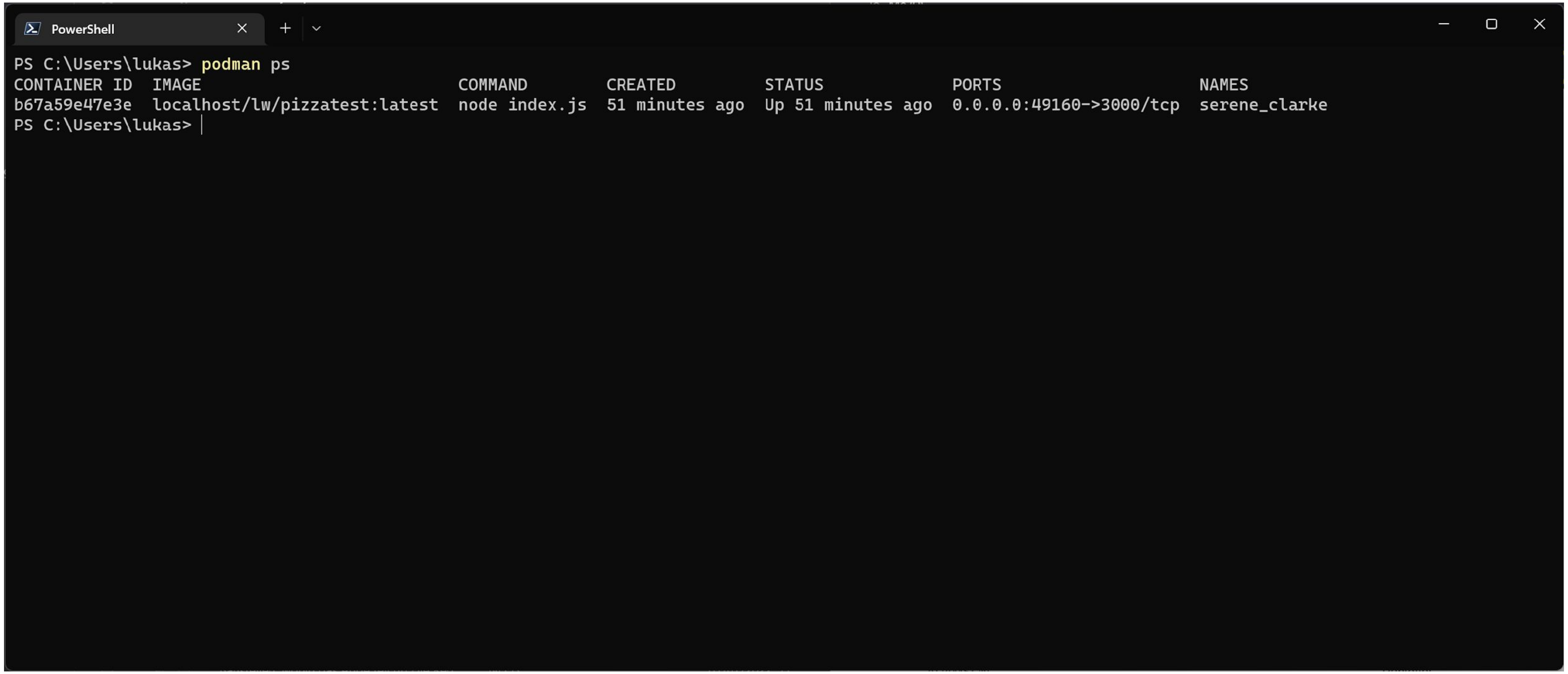
```
podman build . –t lw/pizzatest
```

Start it



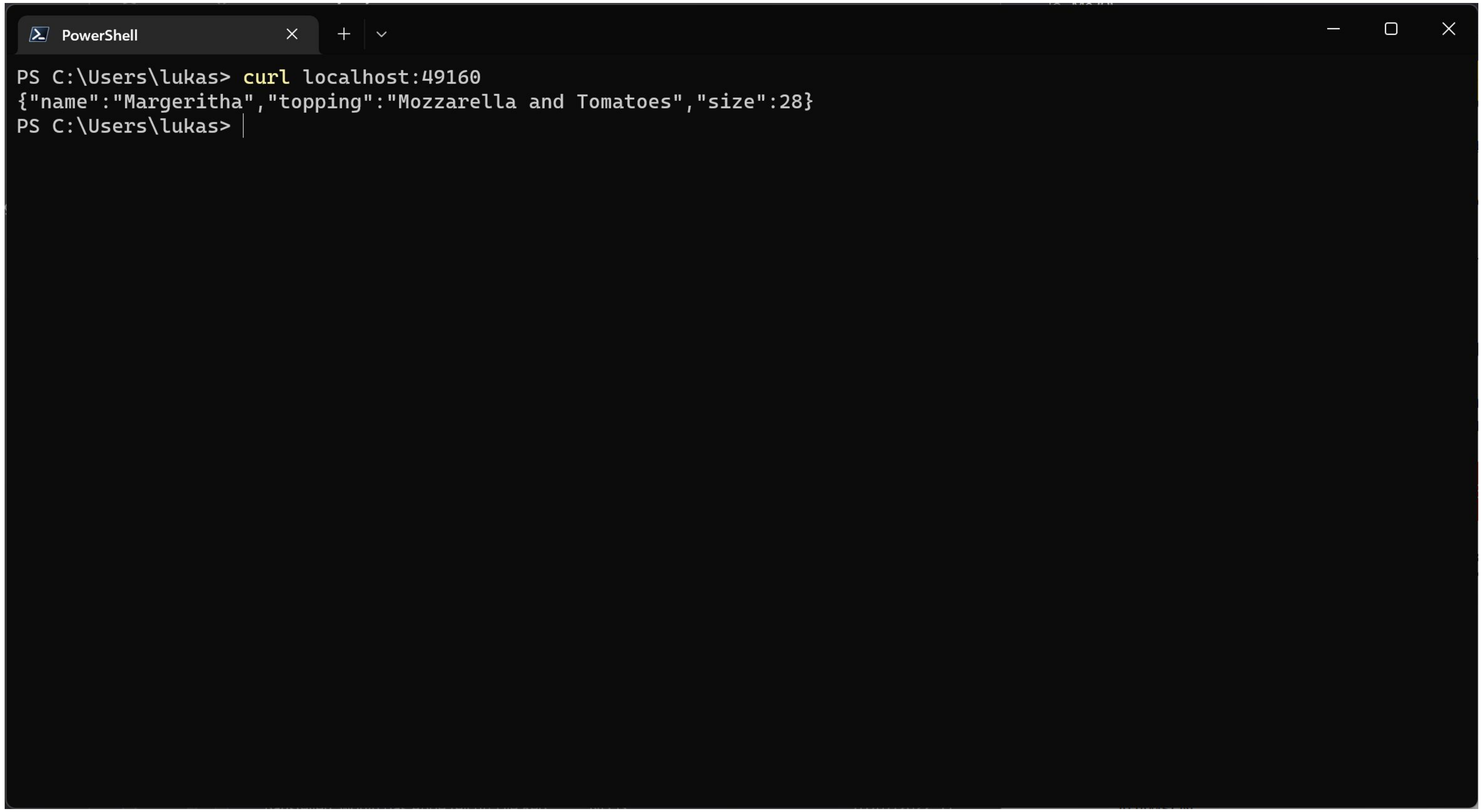
```
podman run -p 49160:3000 -d lw/pizzatest
```

Container is up and running


PowerShell

```
PS C:\Users\lukas> podman ps
CONTAINER ID  IMAGE                          COMMAND        CREATED         STATUS             PORTS                     NAMES
b67a59e47e3e  localhost/lw/pizzatest:latest  node index.js  51 minutes ago  Up 51 minutes ago  0.0.0.0:49160->3000/tcp  serene_clarke
PS C:\Users\lukas>
```

HTTP Request and Response


PowerShell

```
PS C:\Users\lukas> curl localhost:49160
{"name":"Margeritha","topping":"Mozzarella and Tomatoes","size":28}
PS C:\Users\lukas>
```

Most Prominent Container Engines

podman

docker

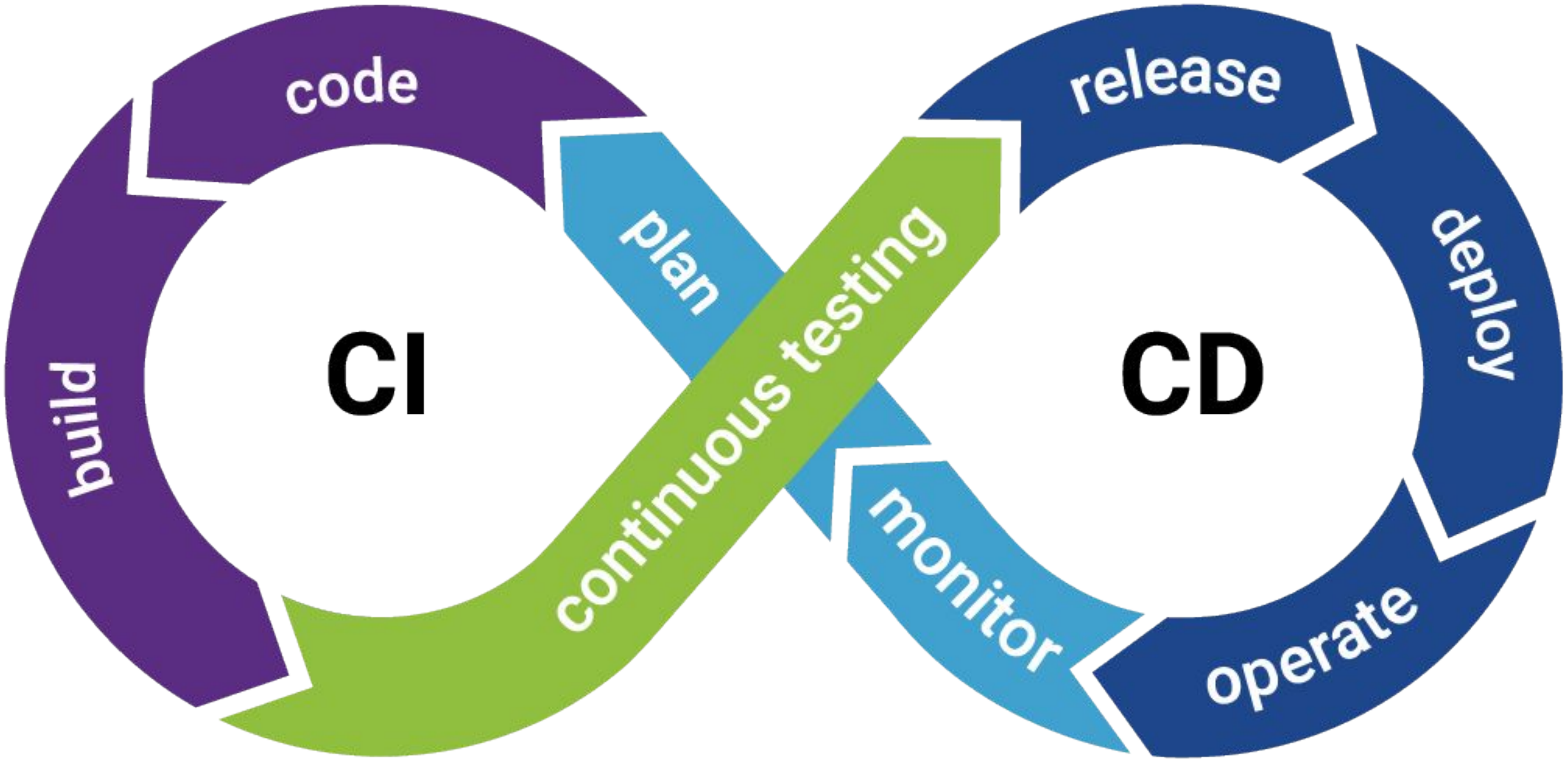
Continuous Integration

Automatically test and deploy your software frequently

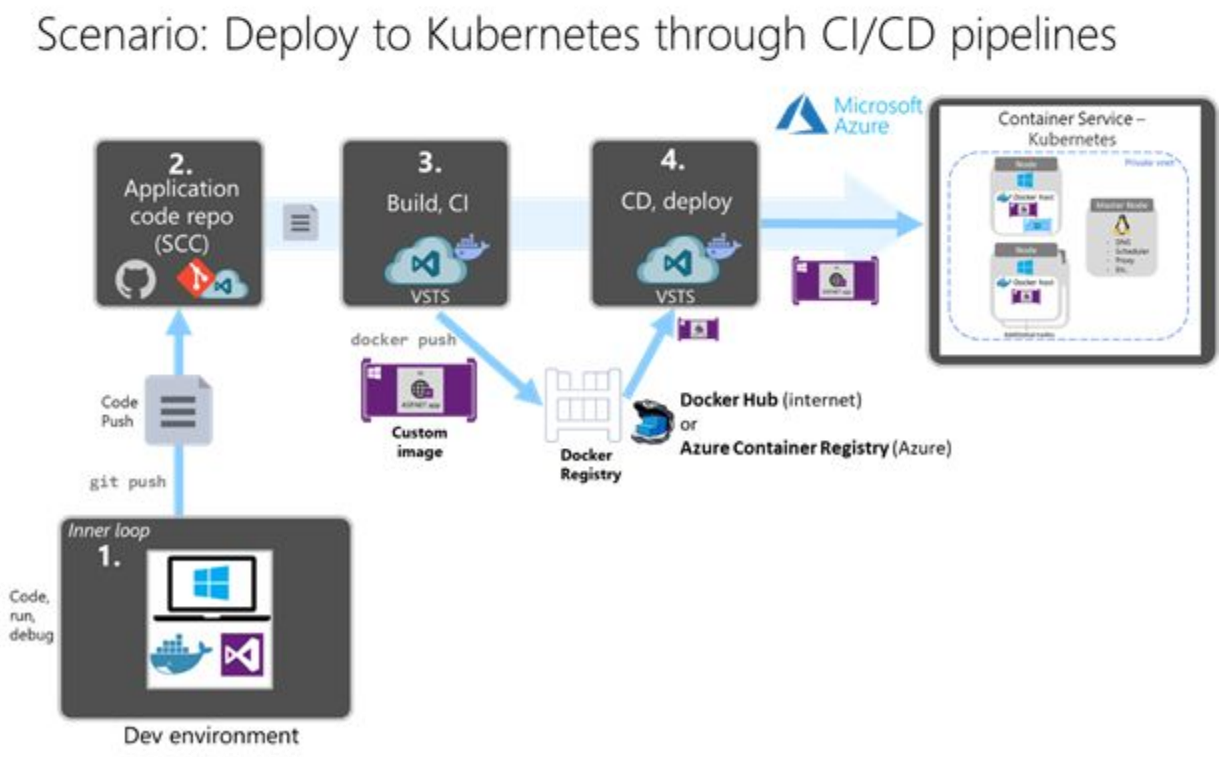
Continuous Deployment

Automatically release and monitor your software

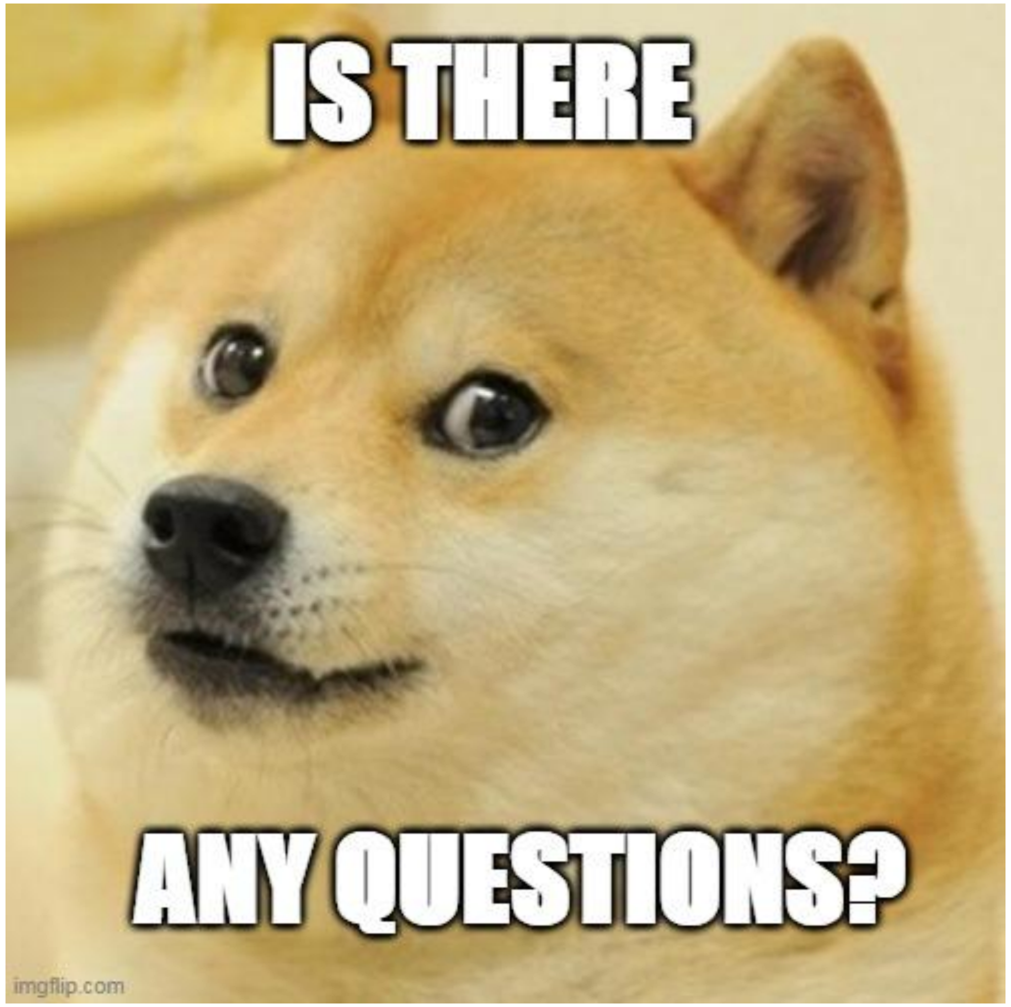
Together: CI/CD-Pipeline



Container in CI/CD-Pipelines


Scenario: Deploy to Kubernetes through CI/CD pipelines
Microsoft Azure
Container Service – Kubernetes
2. Application code repo (SCC)
3. Build, CI
VSTS
4. CD, deploy
VSTS
docker push
Custom image
Docker Registry
Docker Hub (internet) or Azure Container Registry (Azure)
Code Push
git push
Inner loop
1.
Code, run, debug
Dev environment

Source: Erstellen von CI/CD-Pipelines in Azure DevOps Services für eine .NET-Anwendung für Container und anschließendes Bereitstellen in einem Kubernetes-Cluster | Microsoft Docs


IS THERE
ANY QUESTIONS?
imgflip.com