# Docker Overview

## Application Frameworks (SE3040)

Vishan Jayasinghearachchi
Lecturer
Department of Software Engineering, Faculty of Computing
vishan.j@sliit.lk

# Contents

- Docker Overview
  - What is Docker?
  - What does it do?
  - Docker use-case example
  - Docker Architecture
  - Docker Terminology
- Hands-on demo

# Docker Overview

- What is Docker?
  - Docker is a platform that enables packaging and running an application in a loosely isolated environment called a **container**.
  - Docker enables you to separate your applications from your infrastructure.

# Docker Overview

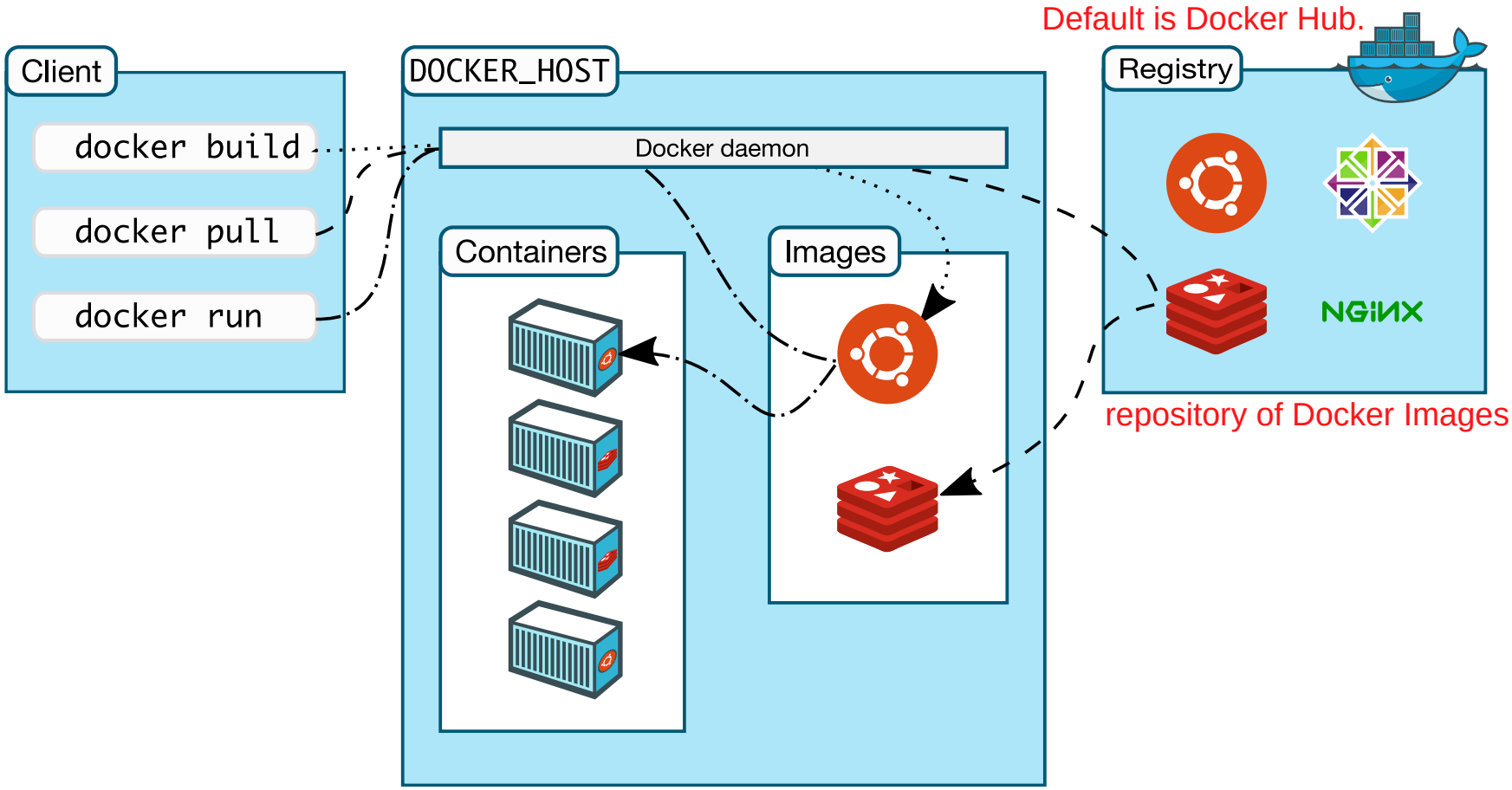
- What does it do?
  - Docker provides tooling and a platform to manage the lifecycle of your containers:
    - Develop your application and its supporting components using containers.
    - The **container** becomes the **unit for distributing and testing** your **application.**
    - When you're ready, deploy your application into your production environment, as a container or an orchestrated service.
    - This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.

# Docker Overview

- Docker use-case example
  - Your developers write code locally and share their work with their colleagues using Docker containers.
  - They use Docker to push their applications into a test environment and execute automated and manual tests.
  - When developers find bugs, they can fix them in the development environment and redeploy them to the test environment for testing and validation.
  - When testing is complete, getting the fix to the customer is as simple as pushing the updated image to the production environment.

# Docker Overview

- Docker Architecture



# Docker Overview

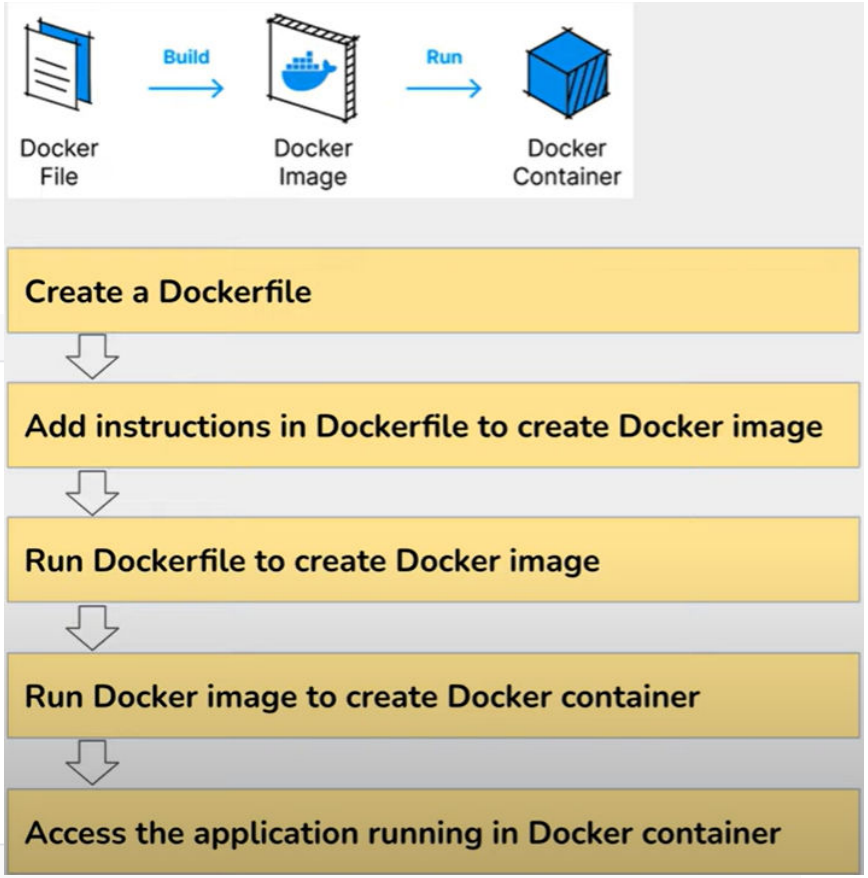
- Docker Architecture
  - Has a Client-Server architecture.
  - **Docker Client** communicates with the **Docker daemon** which handles building and running of the containers.
  - Docker Registry is a repository of **Docker Images**.

# Docker Overview

- Docker Terminology
  - The Docker daemon - Manages Docker objects such as images, containers, networks, and volumes.
  - The Docker client – The program through which users interact with Docker.
  - Docker Desktop – The installation which include the daemon, client, **compose**, **K8s** etc.
  - Docker Registry - Stores Docker images. Default is Docker Hub.
  - Docker Objects
    - Image – A read-only template with instructions for creating a docker container.
    - Container – A runnable instance of an image.
    - Network – Facilitates communication among containers.
    - Volume - Provides the ability to connect specific filesystem paths of the container back to the host machine.
    - Plugin – Third-party extensions to enhance Docker's capabilities.

# Build an image

- Create a Dockerfile.




```
# syntax=docker/dockerfile:1

FROM node:18-alpine
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000
```

A dockerfile is a text file with instruction to build a docker image

Source: https://docs.docker.com/get-started/02_our_app/

when run dockerfile docker image is created

when run docker image, container are created

# Build an image

- Build the Docker Image.
  - *docker build -t getting-started .*

    build a Docker image

Tags the image with the name getting-started

Refers to the current directory, where Docker will look for the Dockerfile

- Create a container from the built Docker Image
  - *docker run -dp 127.0.0.1:3000:3000 getting-started*

Creates and starts a container from the image

- Watch this video demo on building a Docker Image and creating a container using that.

Runs the container in detached mode

Maps your host machine's port 3000 to the container's port 3000, but only allows connections from localhost.

open web browser and navigate http://localhost:3000 inside the container

- The created image can be shared with others via a Container Registry such as the Docker Hub.

# Multi-container apps

- In general, a container should do one thing and do it well.
- A real-world cloud native application based on Microservices will have multiple services implemented on multiple containers.
- Docker Compose is a tool that can be used to define multi-container applications.
  - Watch this video tutorial on Docker Compose.
  - An Example

Docker compose
: tool for defining & running multi-container docker applications
: use yaml files to configure application services (docker-compose.yml)
: can start all services with a single command : docker compose up
: can stop all services with a single command : docker compose down
: can scale up selected services when required

# Self Study

- Play with Docker is an interactive tool to learn Docker via your browser. Try it out.

# Acknowledgements and Additional Reading

- Docker overview