Cloud Computing Applications and Services

(Aplicações e Serviços de Computação em Nuvem)

Guide 2: Docker & Ansible

University of Minho

2026-2027

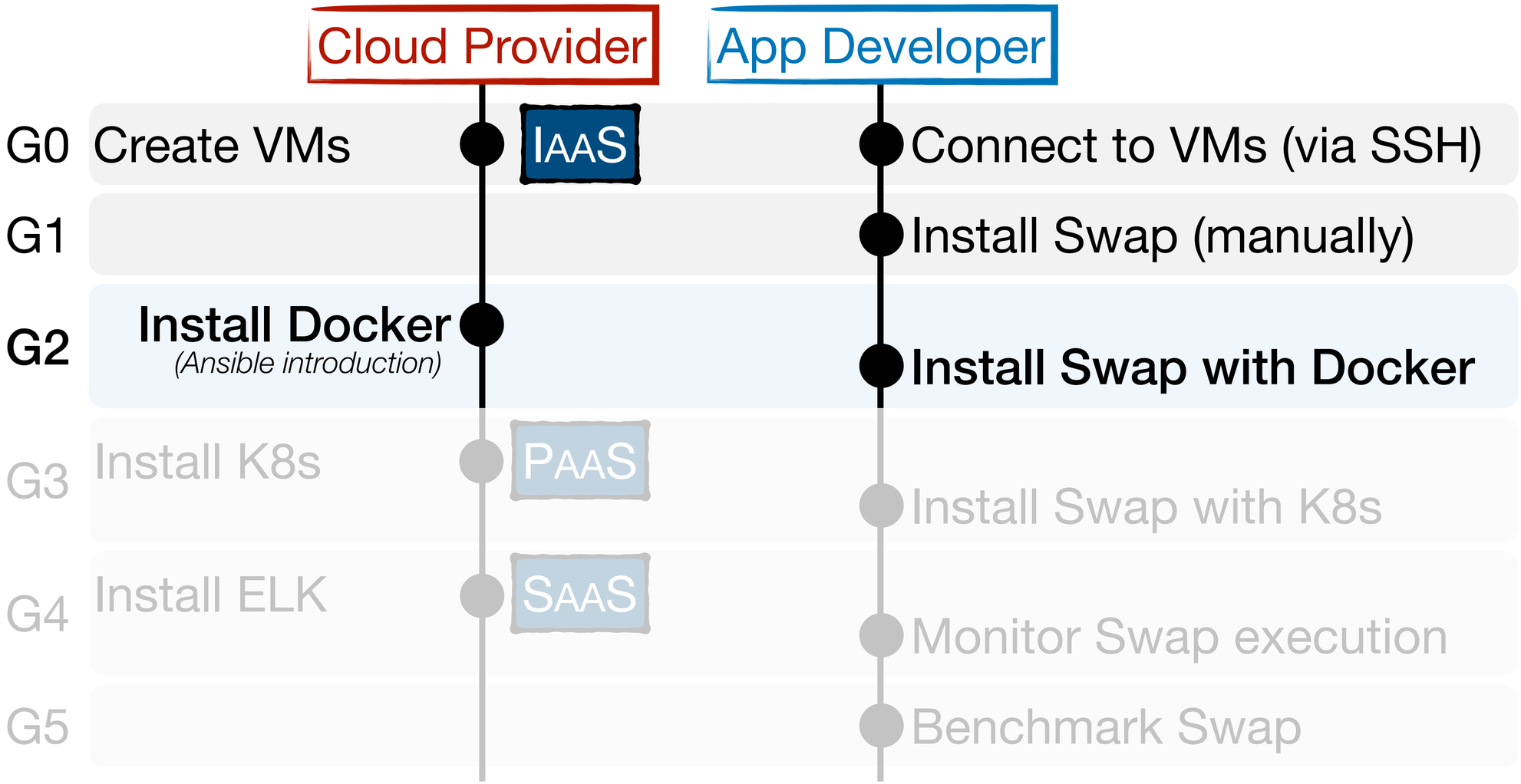


Context

- In Guide 1, you have installed Swap by directly accessing and manually configuring the two VMs “provided by the Cloud”.
- Manually installing an application is a tedious and time-consuming process. You may need to deploy the application several times (e.g., periodical updates). Even worse, imagine that you are asked to deploy *Swap* across all Universities in Portugal!
- Additionally, manual installation is error-prone!
- To address these challenges, you need to **automate the installation and configuration of Swap**!

Road Map

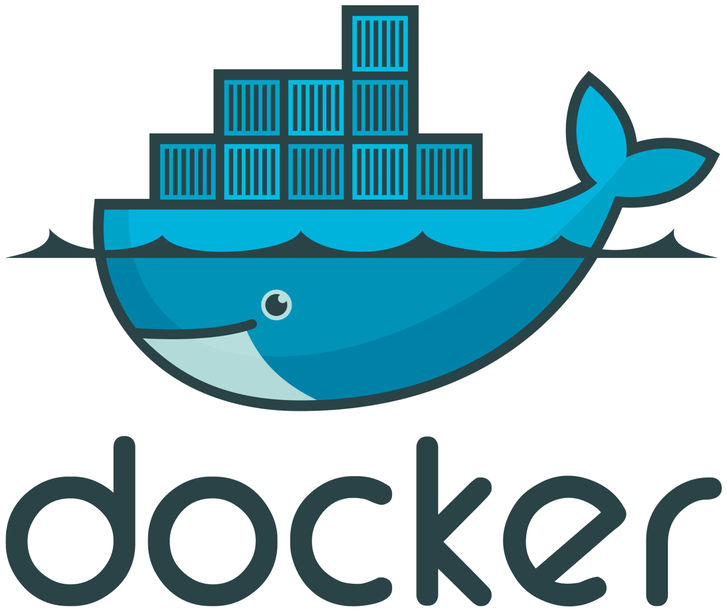
Where are you on the roadmap?



Goal

- In this Guide, you will use tools to automate the **configuration and installation of Swap**.
- The Guide is divided into two parts:
  - **PART I:** Use **Docker** to automate Swap's configuration and component setup.
  - **PART II:** Use **Ansible** to automate the installation of software dependencies (e.g., Docker platform), the deployment of Docker containers, and the exposure of the application to end users.

Part I

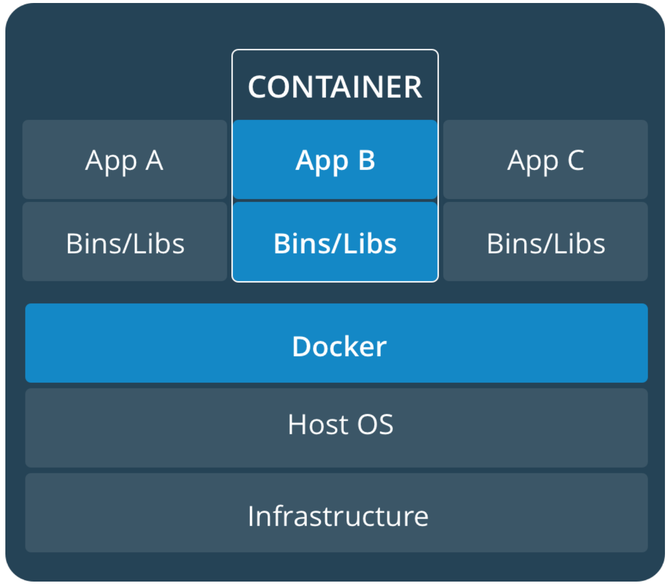


Docker is the most widely-known container technology.

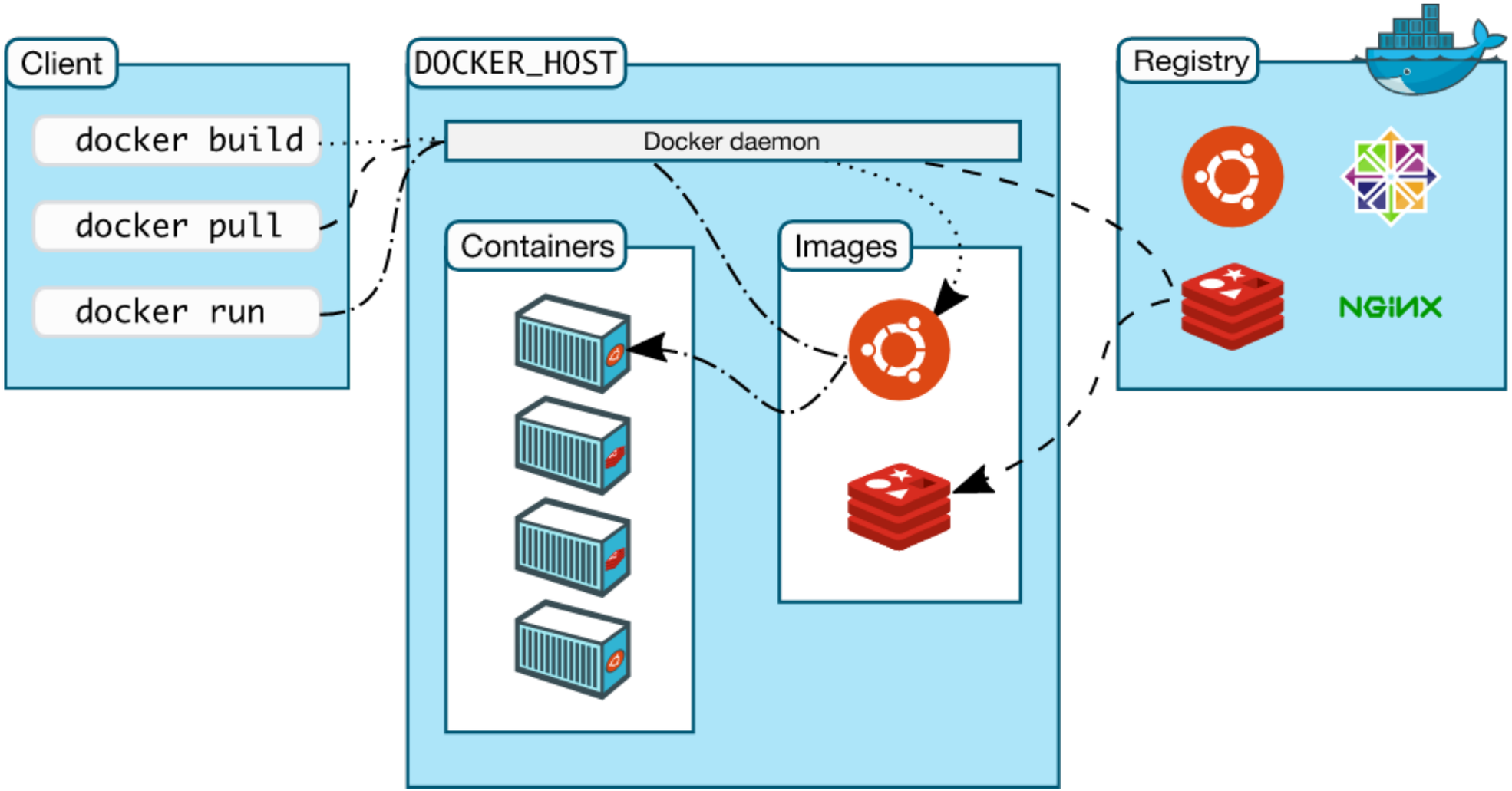
It provides tooling to build, distribute, and run containers.

What is a Container?

- Lightweight virtual environment that groups and isolates a set of processes and resources (RAM, CPU, disk, …), from the host and other containers.
- Why are containers useful?
  - Running different isolated versions of the same software/application (*e.g.*, database) in a shared OS/Kernel environment
  - Portability/migration across servers
  - Easy packaging of software, applications, and their dependencies
- For more info, stay tuned for the theoretical class!



Docker Components



Docker Components

- **Docker Client**
  - Component used by users to interact with the Docker Platform.
- **Docker Daemon**
  - Receives and handles requests from the Docker Client.
  - Manages Docker images, containers, and networks.
- **Docker Image**
  - An immutable file that contains the source code, libraries, and other files needed for an application to run.
- **Docker Container**
  - Runnable instance of an Image.
- **Docker Registry**
  - Repository of Docker Images.

Dockerfile

- A **Dockerfile** is a configuration file that contains all the commands necessary to assemble an image.
- Docker can build images automatically by reading the instructions from a Dockerfile.

```
FROM node:argon

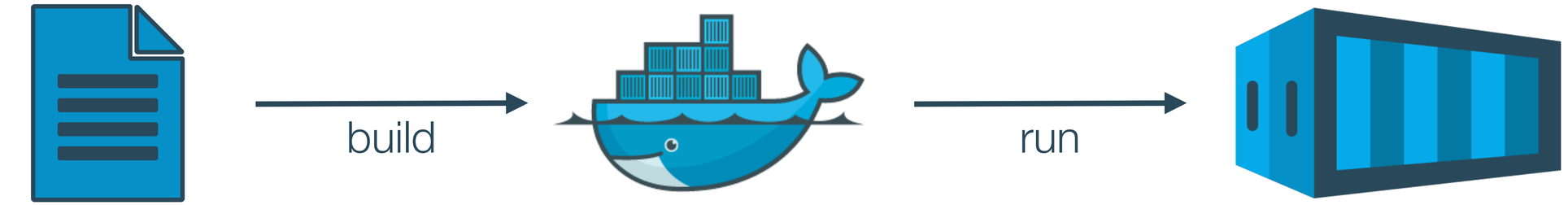
# Create app directory
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

# Install app dependencies
COPY package.json /usr/src/app/
RUN npm install

# Bundle app source
COPY . /usr/src/app

EXPOSE 8080
CMD [ "npm", "start" ]
```

Docker Lifecycle



Dockerfile **Docker Image** **Docker Container**

Building a Docker Image

```
FROM node:argon

# Create app directory
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

# Install app dependencies
COPY package.json /usr/src/app/
RUN npm install

# Bundle app source
COPY . /usr/src/app

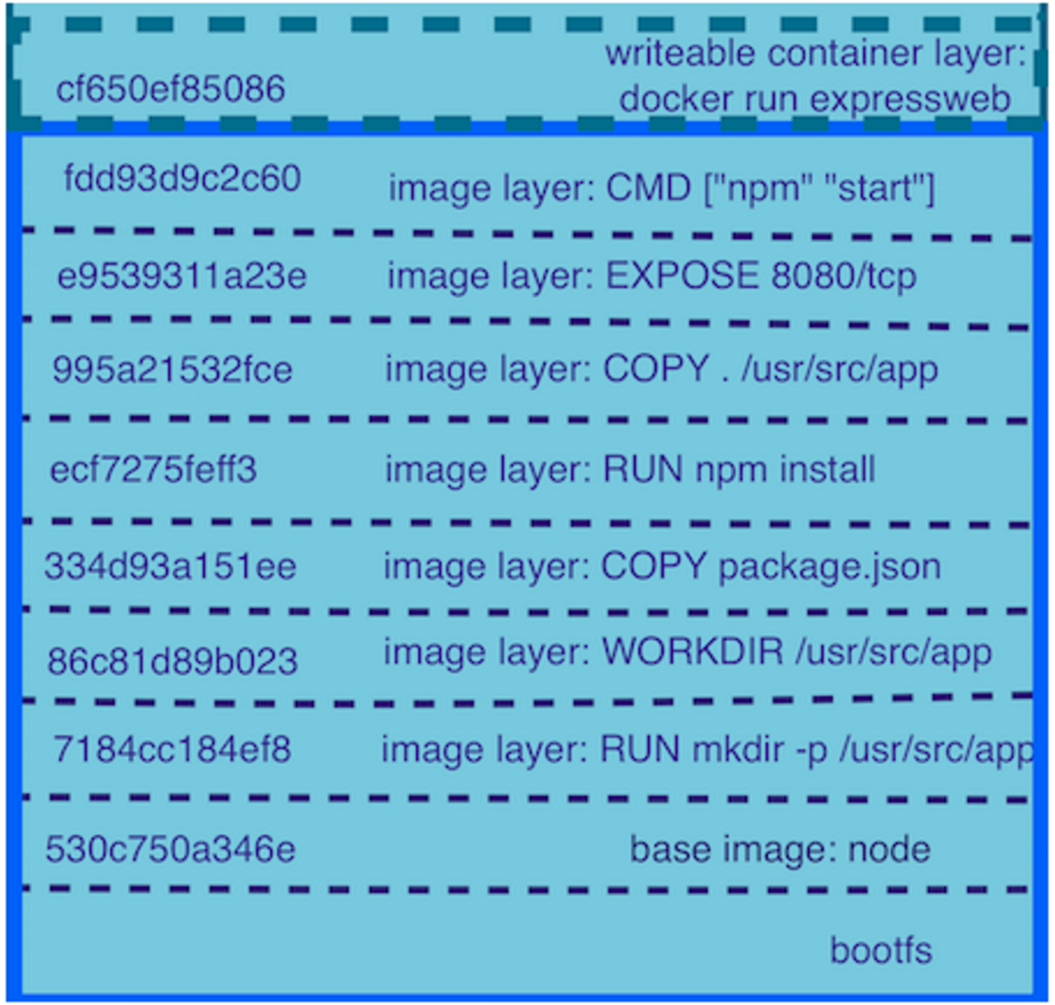
EXPOSE 8080
CMD [ "npm", "start" ]
```



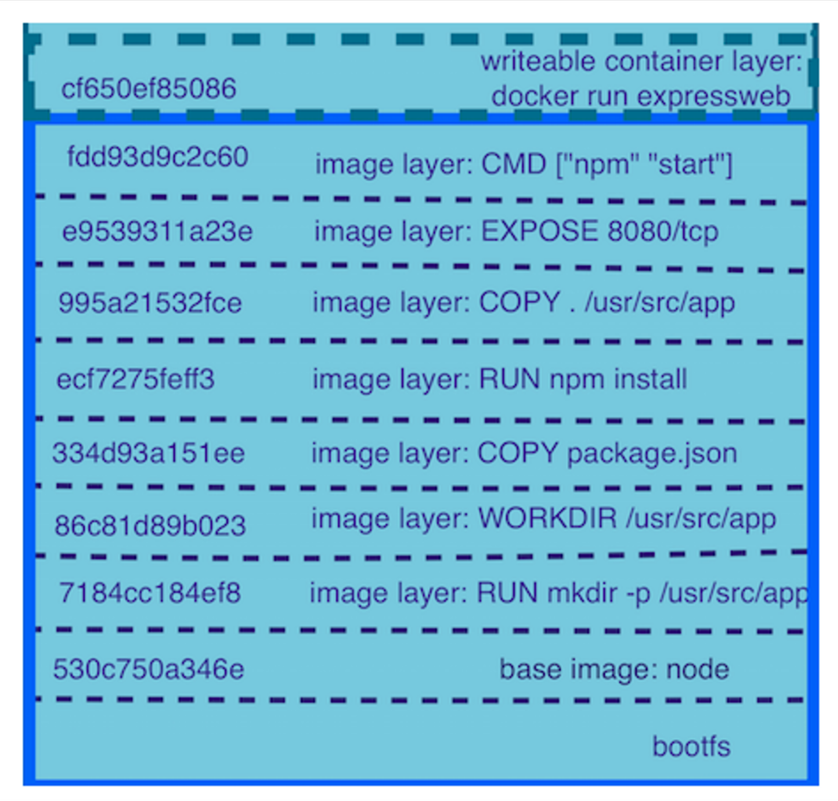

```
$ docker build -t expressweb .
Step 1 : FROM node:argon
argon: Pulling from library/node...
...
Status: Downloaded newer image for node:argon
 ---> 530c750a346e
Step 2 : RUN mkdir -p /usr/src/app
 ---> Running in 5090fde23e44
 ---> 7184cc184ef8
Removing intermediate container 5090fde23e44
Step 3 : WORKDIR /usr/src/app
 ---> Running in 2987746b5fba
 ---> 86c81d89b023
Removing intermediate container 2987746b5fba
Step 4 : COPY package.json /usr/src/app/
 ---> 334d93a151ee
Removing intermediate container a678c817e467
Step 5 : RUN npm install
 ---> Running in 31ee9721cccb
 ---> ecf7275feff3
Removing intermediate container 31ee9721cccb
Step 6 : COPY . /usr/src/app
 ---> 995a21532fce
Removing intermediate container a3b7591bf46d
Step 7 : EXPOSE 8080
 ---> Running in fddb8afb98d7
 ---> e9539311a23e
Removing intermediate container fddb8afb98d7
Step 8 : CMD npm start
 ---> Running in a262fd016da6
 ---> fdd93d9c2c60
Removing intermediate container a262fd016da6
Successfully built fdd93d9c2c60
```

Docker Image

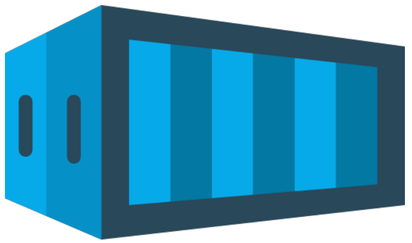
- Docker images consist of read-only layers. When a container is created, Docker adds a writable container layer on top of the image.
- Data persistence must be ensured using bind mounts or volumes!



Running a Container







Docker Container

Persistent Storage

- Store data outside the container's internal file system (Union FS), so it persists even if the container is removed.
- **Bind mount** (-v /host/path:/container/path):
  - Maps a specific host file or directory into the container.
  - Any container or host process can modify this data.
- **Volume** (-v mydata:/container/path):
  - A Docker-managed storage location, independent of the container lifecycle.
  - Unlike bind mounts, it doesn't depend on the host's directory layout and is intended to be accessed through containers.

Find more about Docker Volumes at: https://docs.docker.com/storage/volumes/

Network

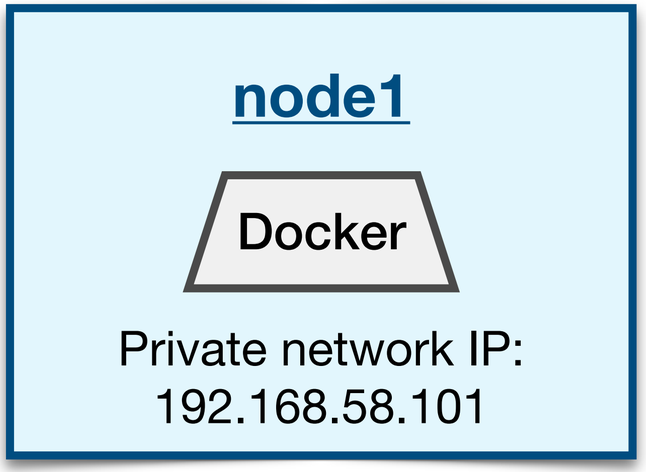
- **Bridge**:
  - Containers are connected to a private virtual network on the host.
  - Containers on the same bridge can communicate with each other. On user-defined bridges, containers can reach each other by name.
  - Services can be exposed outside the host by publishing ports (e.g., `docker run -p 8080:80 …`)
- **Host**:
  - The container shares the host's network namespace.
  - No network isolation between the container and the host.
  - The container uses the host's ports directly (e.g., port 80).

Find more about Docker Networks at: https://docs.docker.com/network/

Part I - Goal

Cloud Provider

- Install and configure the Docker service on node1

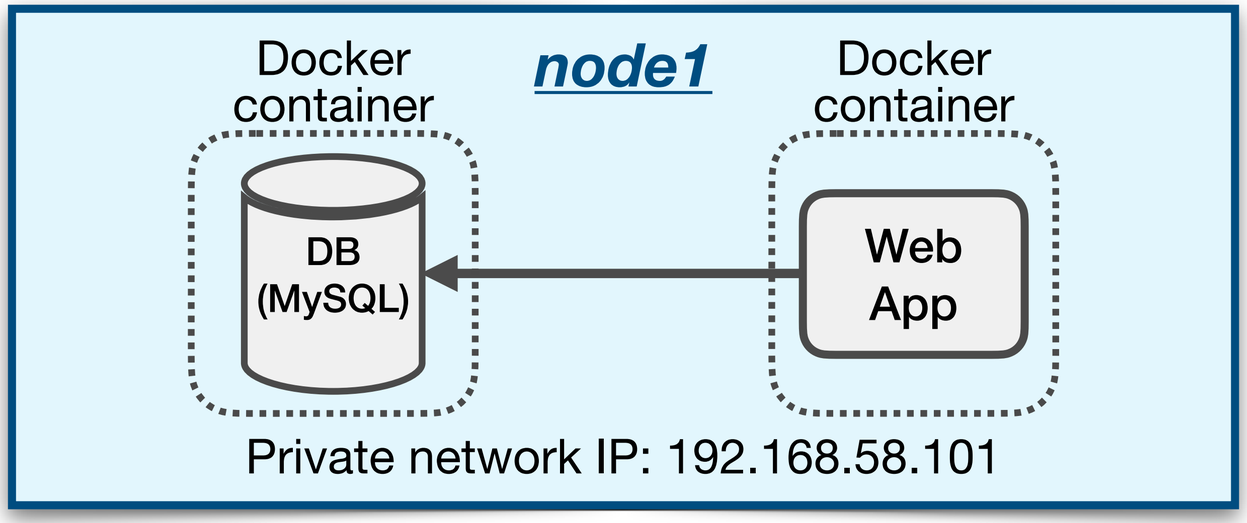


Part I - Goal

App Developer

Now that the Cloud Provider offers a Docker environment:

- Deploy MySQL and Swap containers (on node1) with Docker



Docker
Useful Commands

- **Build** a Docker Image:
  - docker build -t <image_name> .
- **List** Docker Images:
  - docker image ls
  - docker images
- **Delete** a Docker Image:
  - docker image rm <image_name>
  - docker rmi <image_name>
- **Run** a Docker Container:
  - docker run --name <container_name> [options] <image_name> [cmd]
- **List** Docker Containers
  - docker ps [-a]
- Check Docker **logs**
  - docker logs <container_name>
- **Delete** a Docker Container
  - docker rm <container_name>

Part II

Part II - Context

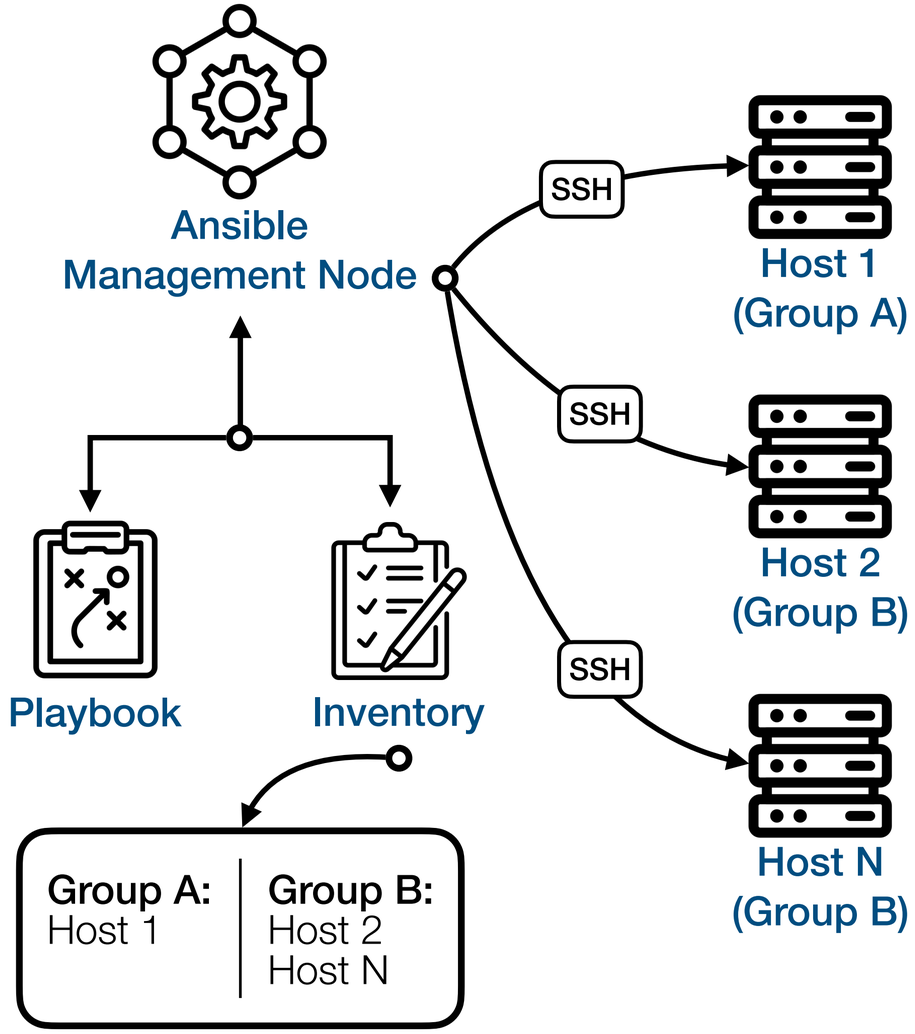
- Now, installing Swap is simplified with Docker.
- However, what if we wanted to deploy Swap on **several machines**? (Remember that you need to do this for all Portuguese Universities…)
  - You would have to:
    - manually install Docker on all of those machines (assuming that their infrastructure does not have Docker installed)
    - manually deploy Swaps's containers on each machine
- You would benefit from further automating these manual steps.
- **Ansible** will help you with this!



Ansible is an open-source automation tool used for configuration management, application deployment, intra-service orchestration, and provisioning.

How Ansible works

- Ansible is operated from a **Management Node**, where you write and execute your Ansible playbooks and commands.
- The list of hosts to be managed by Ansible is specified in the **Inventory** file.
- Ansible connects to remote hosts via SSH and executes the tasks defined in a **Playbook**.



Vocabulary

- **Inventory**
  - Grouped deployment targets (hosts)
- **Module**
  - Reusable work unit distributed with Ansible or developed for it
- **Task**
  - A combination of a module and given arguments in order to create an action
- **Handlers**
  - A special kind of task that responds to a notification
- **Templates**
  - Enable the creation of dynamic configuration (leverages Jinja2, the Python template engine)
- **Role**
  - Reusable component that encapsulates variables, templates, tasks, handlers… (configurable)
- **Playbook**
  - Describe policies for remote systems to enforce (set of roles/tasks)

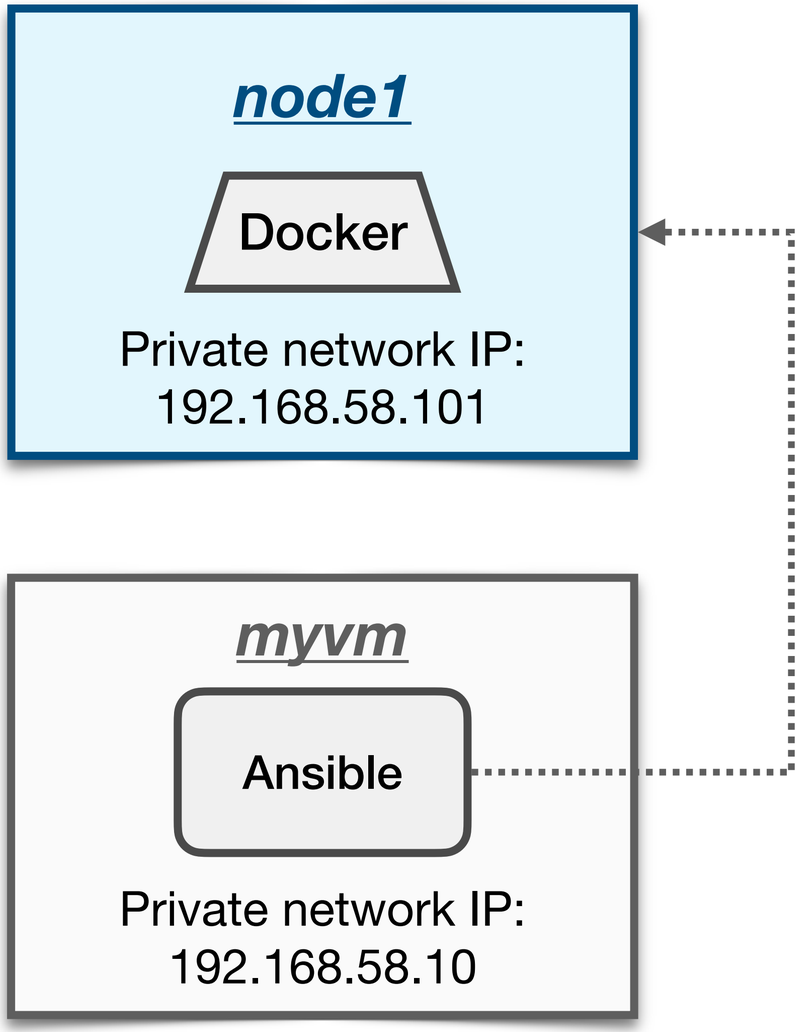
Ansible
Useful Commands

- **Check** connectivity of hosts:
  - ansible <group> -m ping
- **Run** a Playbook:
  - ansible-playbook <playbook>
- List all **tasks** in a Playbook:
  - ansible-playbook <playbook> --list-tasks
- List all **tags** in a Playbook:
  - ansible-playbook <playbook> --list-tags
- Run only plays and tasks with a specific tag:
  - ansible-playbook <playbook> -t <tag>
- Run all plays and tasks except the ones with a specific tag:
  - ansible-playbook <playbook> --skip-tags <tags>

Part II - Goal

Cloud Provider

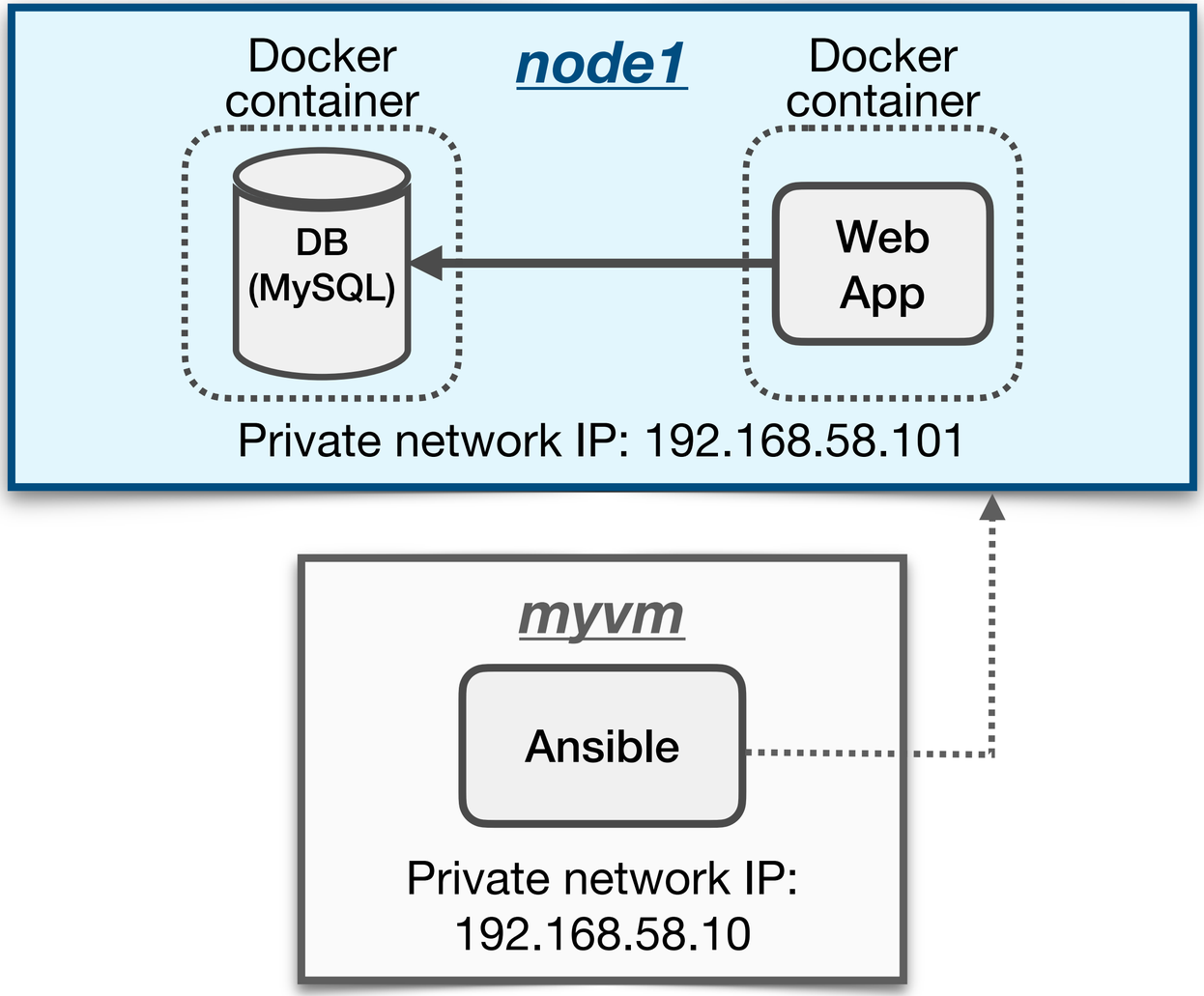
- Use Ansible to automate the installation and configuration of Docker on *node1.*



Part II - Goal

App Developer

- Use Ansible to automate the deployment of Swap with Docker.



Part II - Material provided

This guide is accompanied by two Ansible projects.
Download and inspect them as you work through the practical guide.

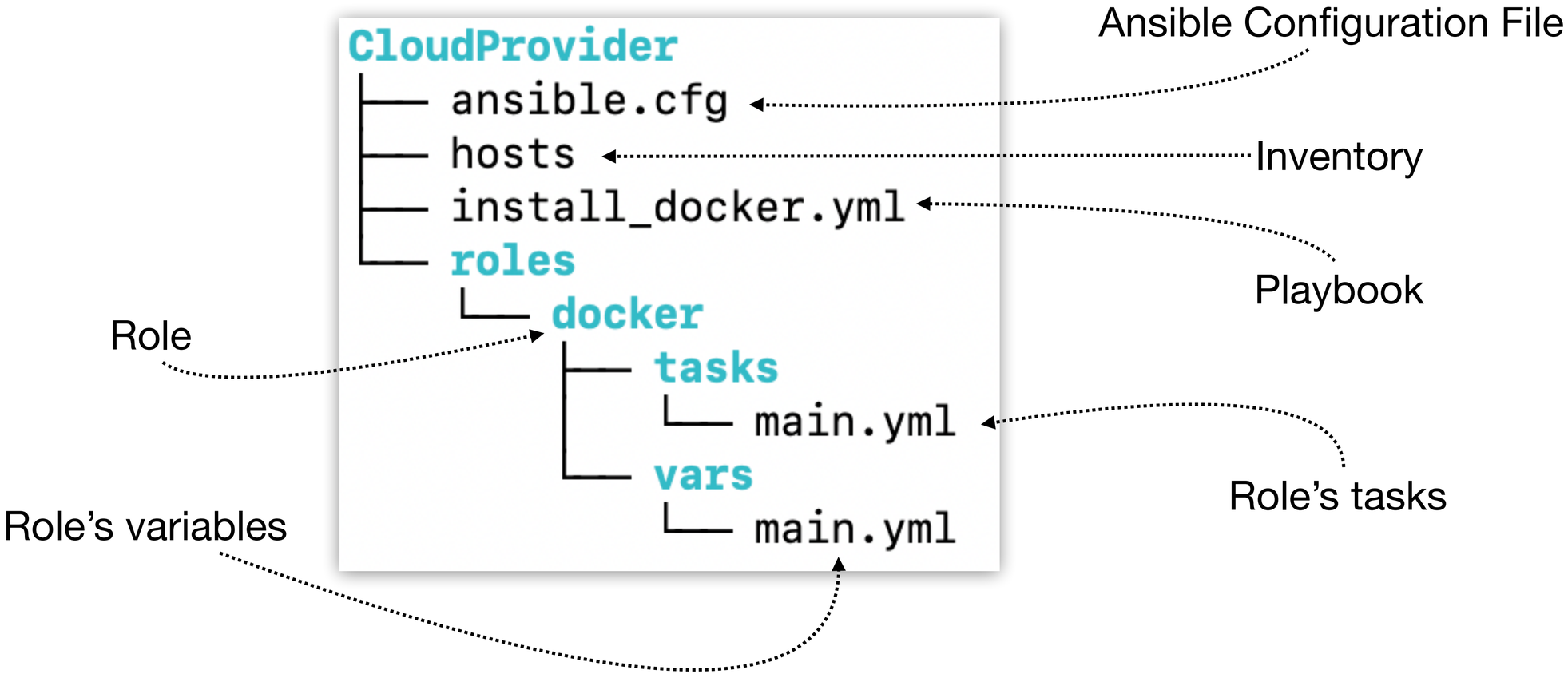
Cloud Provider

- Cloud Provider Project (*CloudProvider* folder):
  - Ansible project to be used by **cloud providers** to provision the cloud infrastructure

App Developer

- App developer Project (*AppDeveloper* folder):
  - Ansible project to be used by **App developers** to install and configure applications on the cloud infrastructure

CloudProvider Project



CloudProvider Project

Inventory (hosts)

```
[nodes]
node1 ansible_host=192.168.58.101
node2 ansible_host=192.168.58.102
```

Group's name

Name and IP of the group's hosts

Playbook (install_docker.yml)

```
- name: Install Docker
  hosts: nodes
  roles:
    - role: docker
      tags: ["docker"]
```

Name of the Play

Role to execute on the targeted hosts

Targeted group of hosts

CloudProvider Project

Role's tasks
(roles/docker/tasks/main.yml)

Task's name

```yaml
- name: Update apt cache
  become: true
  ansible.builtin.apt:
    update_cache: true

- name: Install required packages
  become: true
  ansible.builtin.apt:
    pkg: "{{ docker_dependencies_pkg }}"
```

The become directive allows for executing the task with sudo permissions

Ansible module

Role's variables
(roles/docker/vars/main.yml)

```yaml
docker_dependencies_pkg:
  - apt-transport-https
  - ca-certificates
  - curl
  - gnupg
  - software-properties-common
```

docker_dependencies_pkg is variable defined in the vars/main.yml file and used in the task.

AppDeveloper Project

