Cloud Computing Applications and Services

(Aplicações e Serviços de Computação em Nuvem)

Guide 2: Docker & Ansible

University of Minho

2024-2025

Context

- In Guide 1, you have installed Swap 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 also error prone!
- To address these challenges you need to **automate the installation and configuration of Swap**!

Road Map

Where are you on the roadmap?

	Cloud Provider		App Developer
G0	Create VMs	IAAS	Connect to VMs (via SSH)
G1			Install Swap (manually)
G2	**Install Docker** *(Ansible introduction)*		**Install Swap with Docker**
G3	Install K8s	PAAS	Install Swap with K8s
G4	Install ELK	SAAS	Monitor Swap execution
G5			Benchmark Swap

Goal

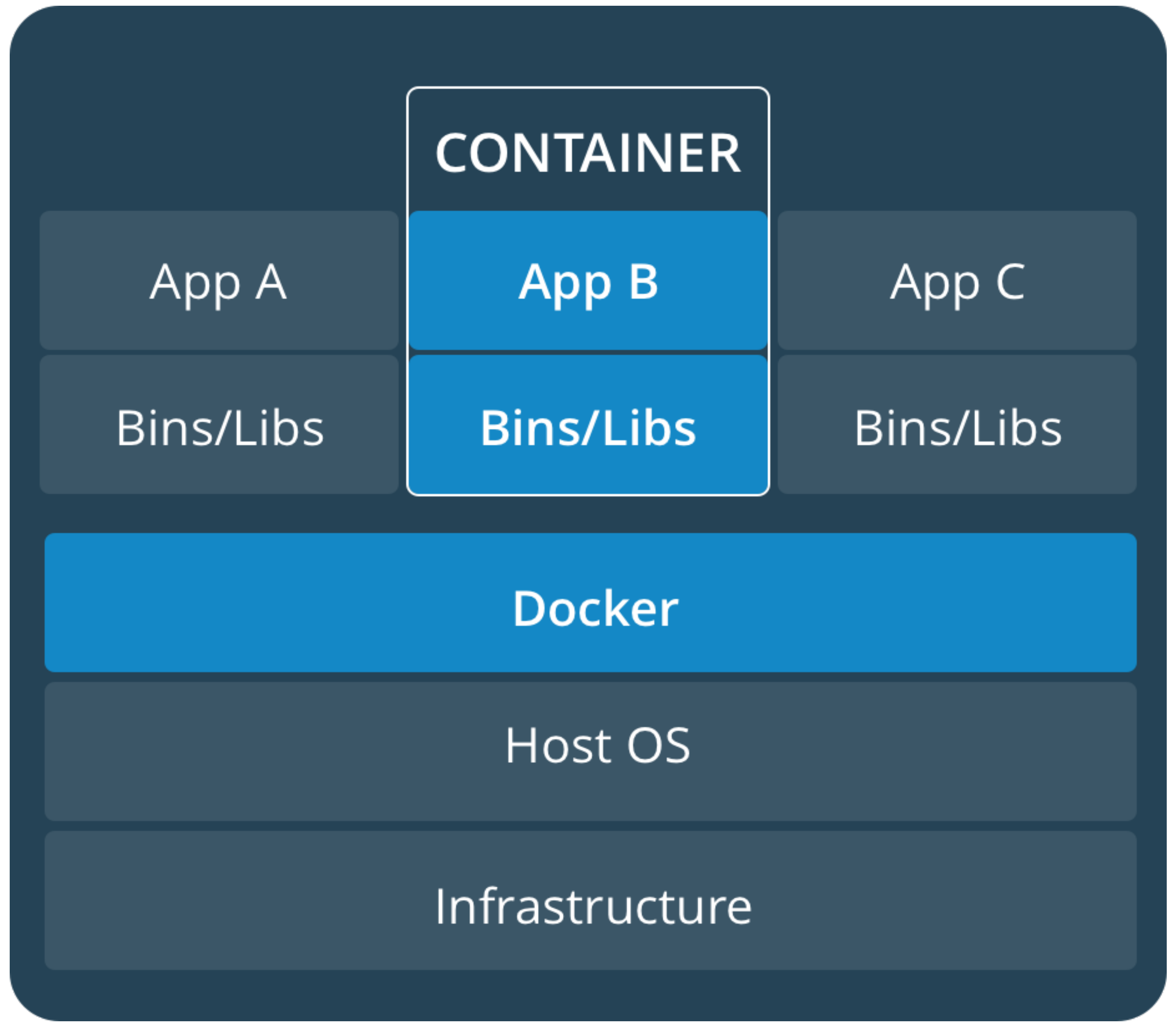
- In this Guide you will use tools for the **automated configuration and installation of Swap**.
- The Guide is divided in two parts:
  - **PART I:** use **Docker** to automate Swap's configuration and component setup.
  - **PART II:** use **Ansible** to automate the installation of software dependencies (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.

What is a Container?

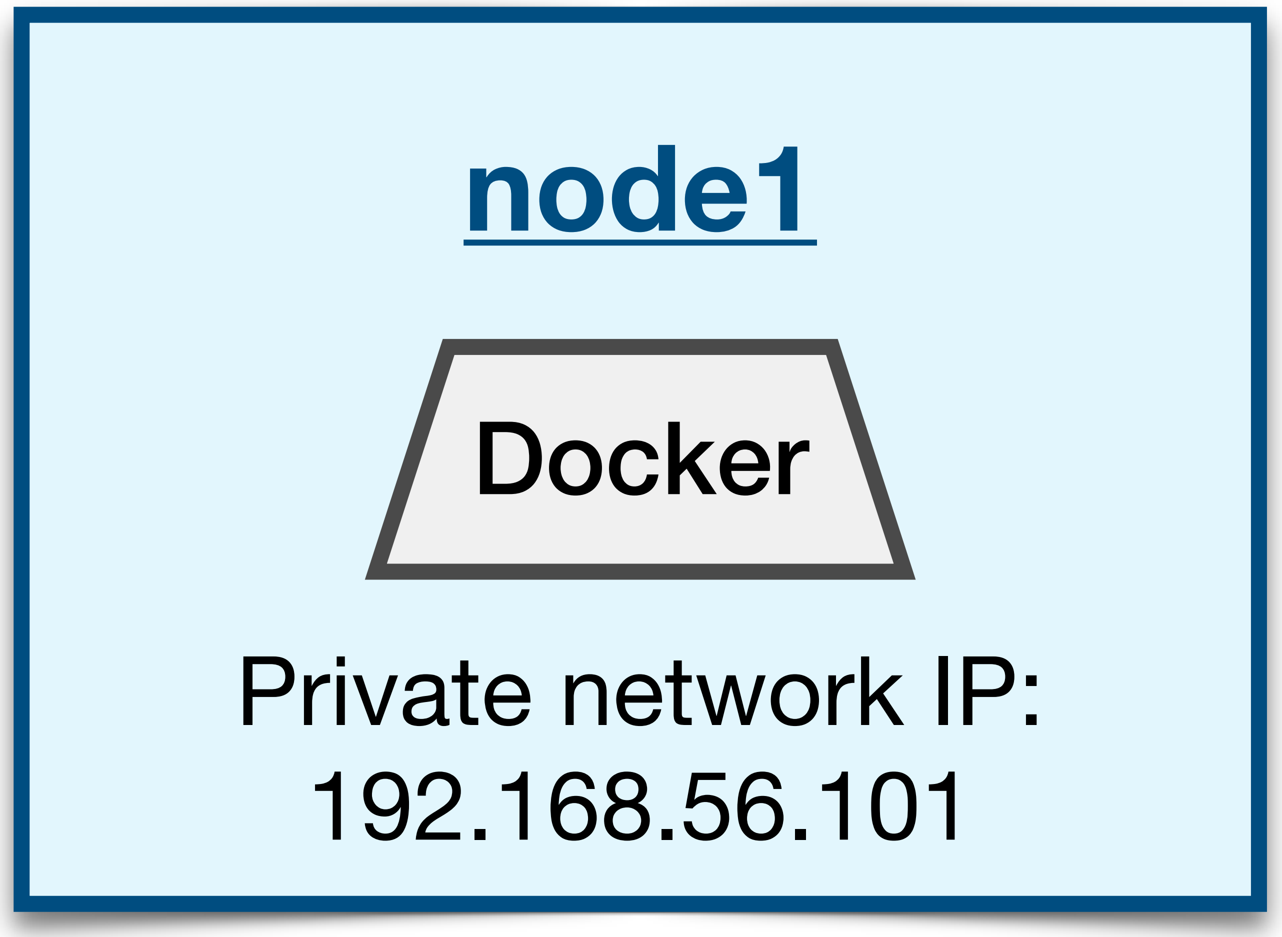
- Lightweight virtual environment that groups and isolates a set of processes and resources (RAM, CPU, disk, …), from the host and other containers.
- Why are containers useful?
  - Running different isolated versions of the same software/application (*e.g.*, database) in a shared OS/ Kernel environment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



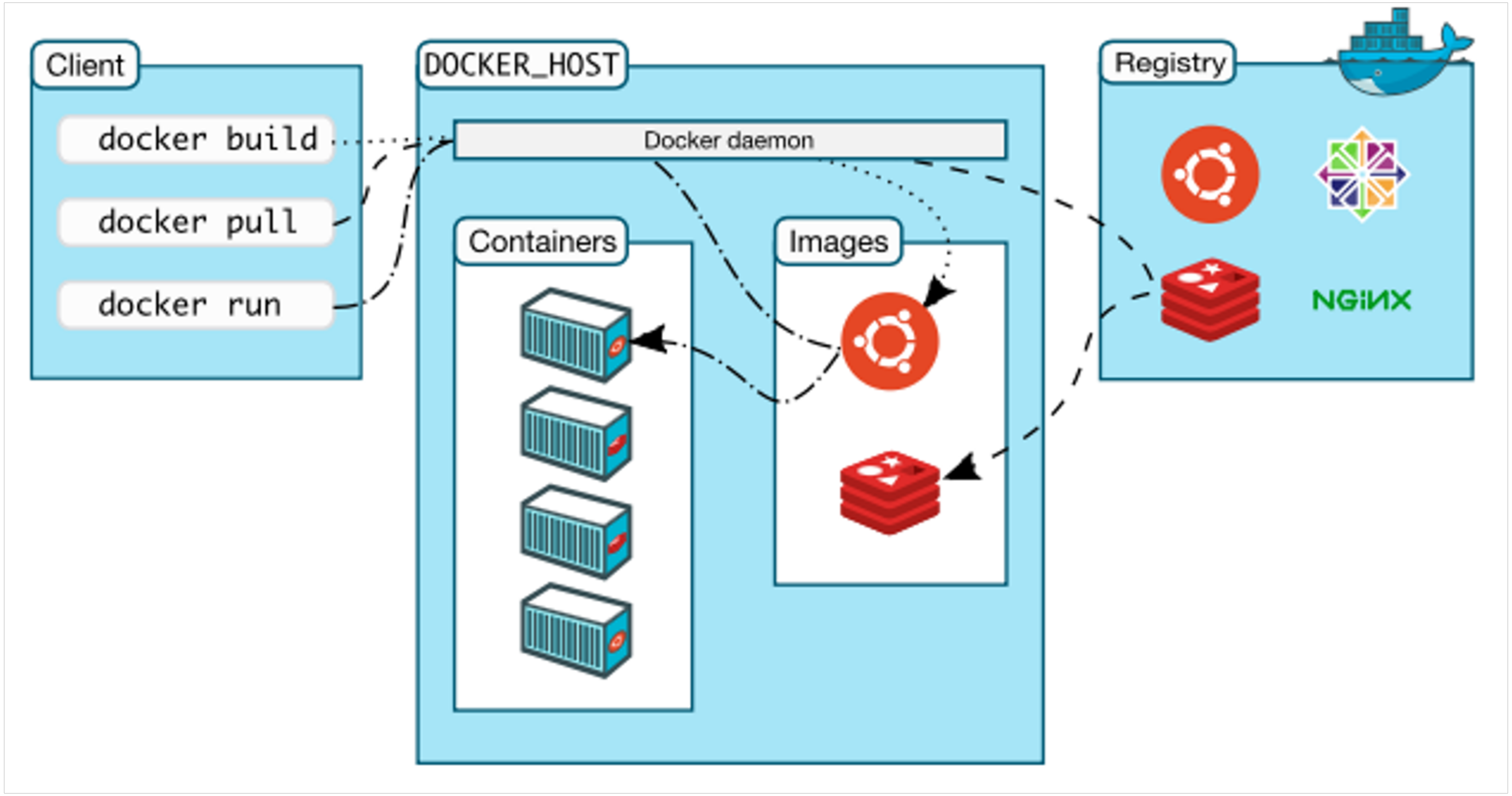
Part I - Goal

Cloud Provider

- Install and configure the Docker service on node1



Docker Components

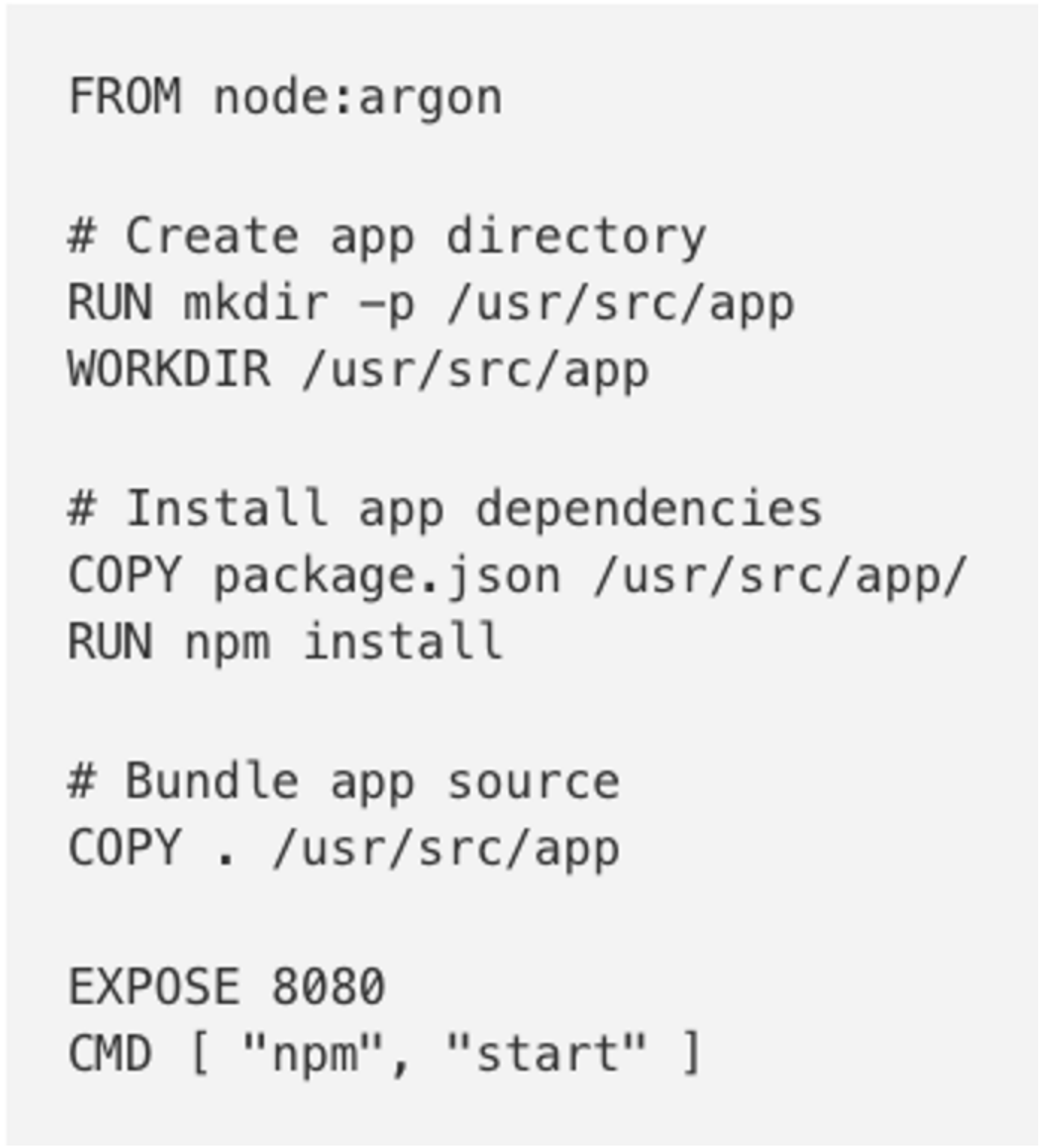


Docker Components

- ◉ **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**
  - ‣ 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.

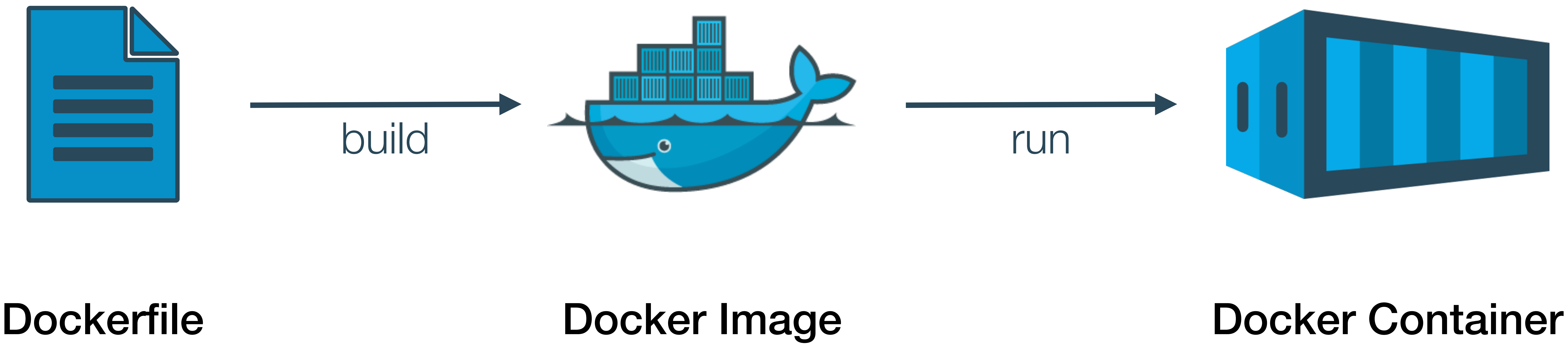
Dockerfile

- **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



Building a Docker Image

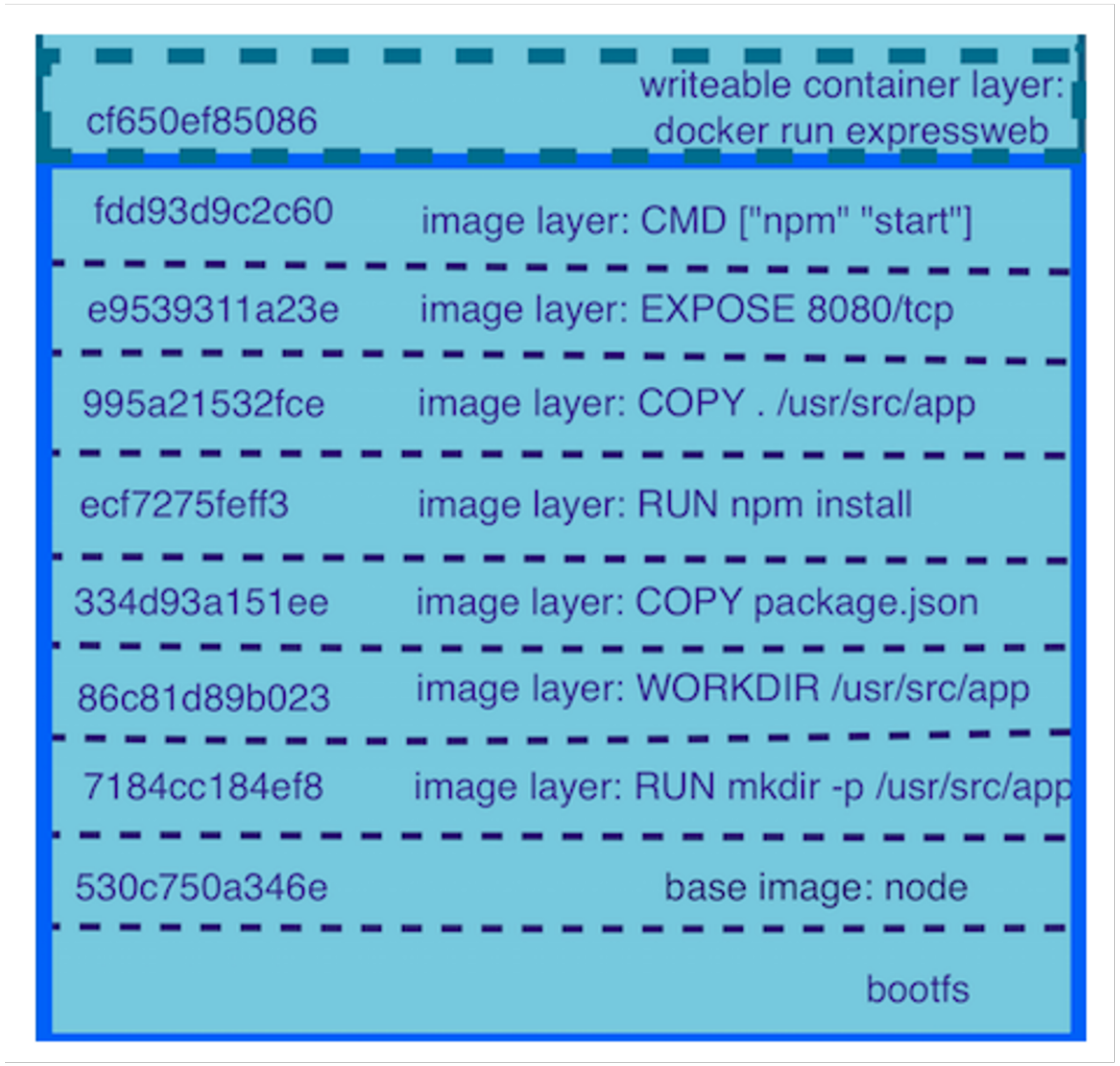
```
FROM node:argon

# 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

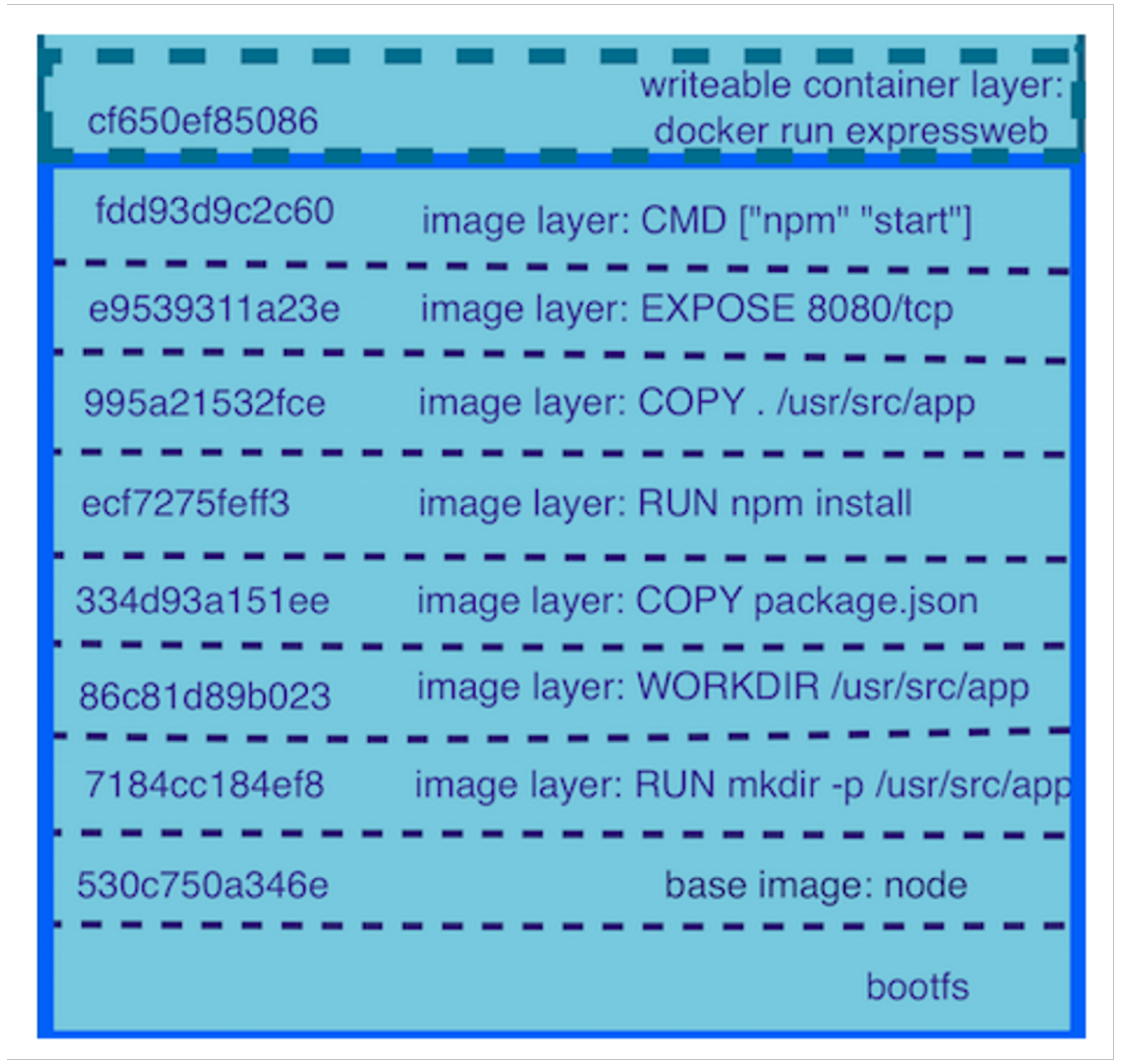
```
$ docker build -t expressweb .
Step 1 : FROM node:argon
argon: Pulling from library/node...
...
Status: Downloaded newer image for node:argon
 ---> 530c750a346e
Step 2 : RUN mkdir -p /usr/src/app
 ---> Running in 5090fde23e44
 ---> 7184cc184ef8
Removing intermediate container 5090fde23e44
Step 3 : 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

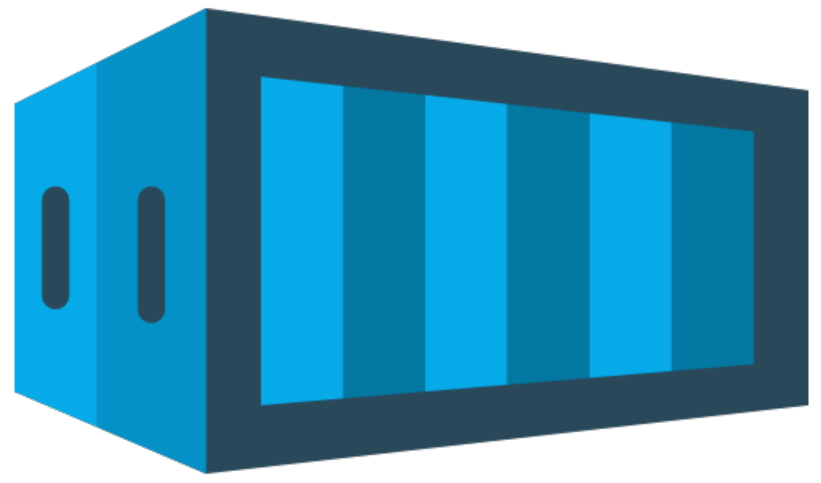
- The first layer is writable while the other are read-only.
- Data persistency needs to be ensured with bind mounts or volumes!


cf650ef85086
writeable container layer: docker run expressweb
fdd93d9c2c60
image layer: CMD ["npm" "start"]
e9539311a23e
image layer: EXPOSE 8080/tcp
995a21532fce
image layer: COPY . /usr/src/app
ecf7275feff3
image layer: RUN npm install
334d93a151ee
image layer: COPY package.json
86c81d89b023
image layer: WORKDIR /usr/src/app
7184cc184ef8
image layer: RUN mkdir -p /usr/src/app
530c750a346e
base image: node
bootfs

Running a Container


cf650ef85086
writeable container layer: docker run expressweb
fdd93d9c2c60
image layer: CMD ["npm" "start"]
e9539311a23e
image layer: EXPOSE 8080/tcp
995a21532fce
image layer: COPY . /usr/src/app
ecf7275feff3
image layer: RUN npm install
334d93a151ee
image layer: COPY package.json
86c81d89b023
image layer: WORKDIR /usr/src/app
7184cc184ef8
image layer: RUN mkdir -p /usr/src/app
530c750a346e
base image: node
bootfs

docker run



Docker Container

Persistent Storage

- Mount a file or directory from the host machine at the container. Stored data is independent from the container's internal file system (Union FS) and persisted even if the container is removed.
- __Bind mount__:
  - Generic directory from the host machine.
  - Any container or host process can access this data.
- __Volume__:
  - A special directory in the host that is managed by Docker and only accessible by containers.

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

Network

- **Host**:
  - Shares the host networking namespace.
  - Container services are presented in the network as if run by the host.
  - Ports are shared (e.g., port 80).
- **Bridge**:
  - The container is seen as another node in the physical network.

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

Docker
Useful Commands

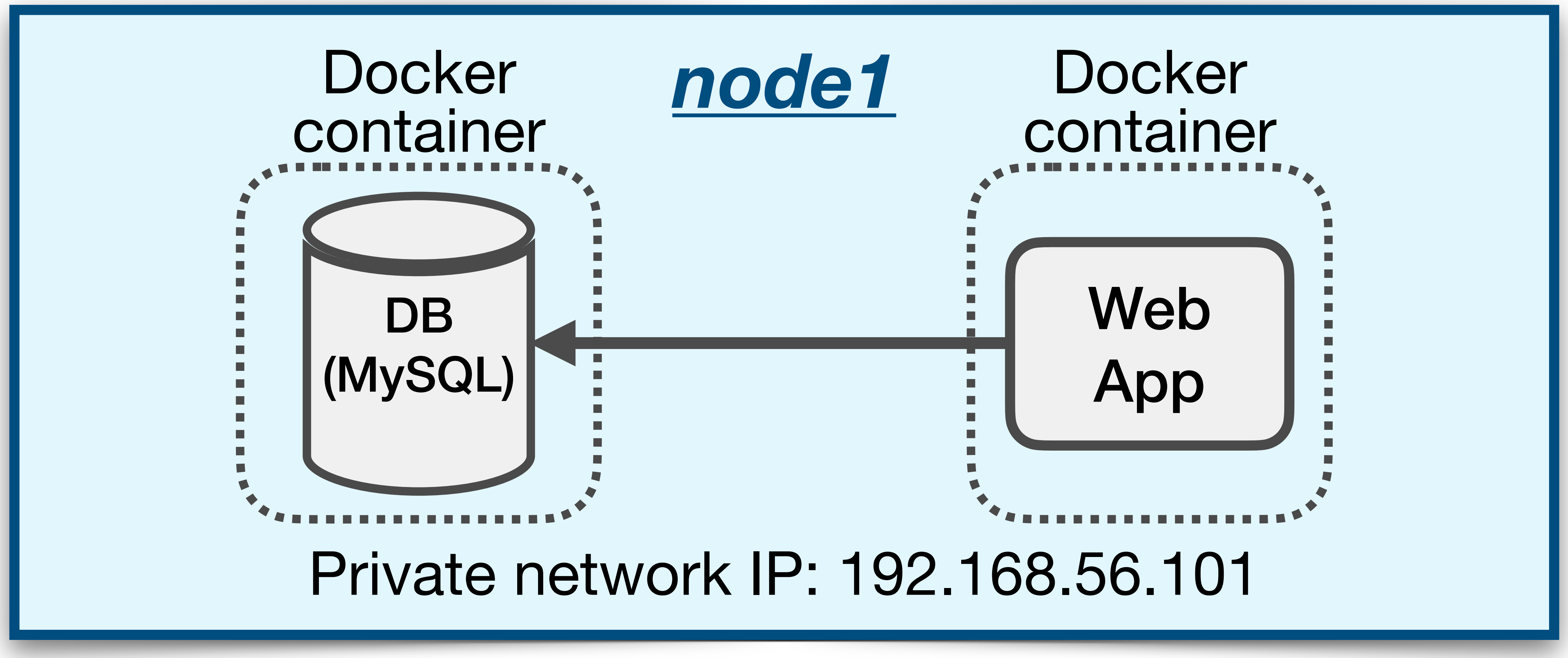
- **<u>Build</u>** a Docker Image:
  - docker build -t <image_name> .
- **<u>List</u>** Docker Images:
  - docker image ls
  - docker images
- **<u>Delete</u>** a Docker Image:
  - docker image rm <image_name>
  - docker rmi <image_name>
- **<u>Run</u>** a Docker Container:
  - docker run --name <container_name>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

Part I - Goal

App Developer

Now that the Cloud Provider offers a docker environment:

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



Part II

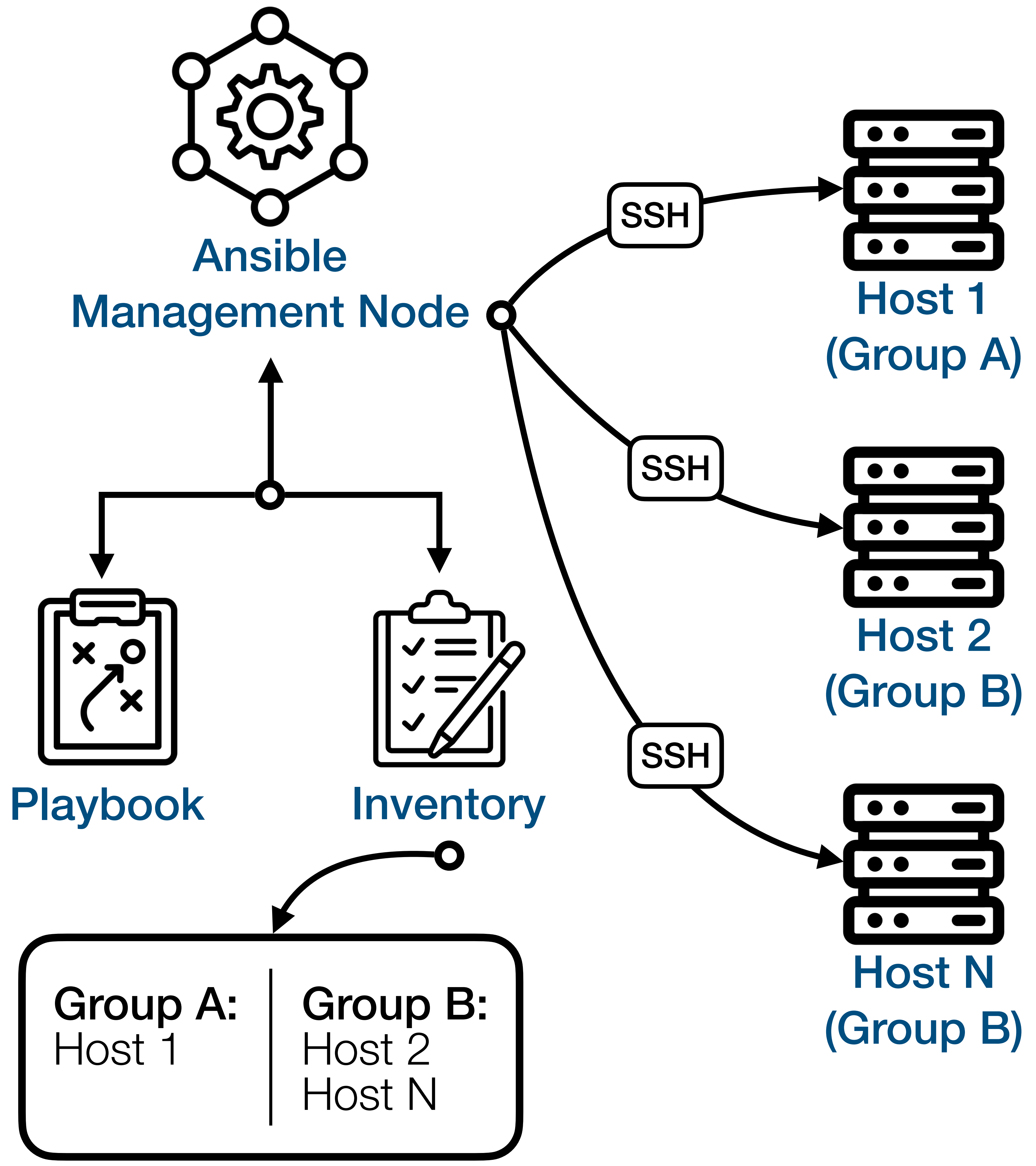
Part II - Context

- Now, your installation of 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 <u>**Management Node**</u>, where you write and execute your Ansible playbooks and commands.
- The list of hosts to be managed by Ansible is specified in the <u>**Inventory**</u> file.
- Ansible connects to remote hosts using SSH and executes the set of tasks defined in a <u>**Playbook**</u>.



Ansible Vocabulary

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

Ansible

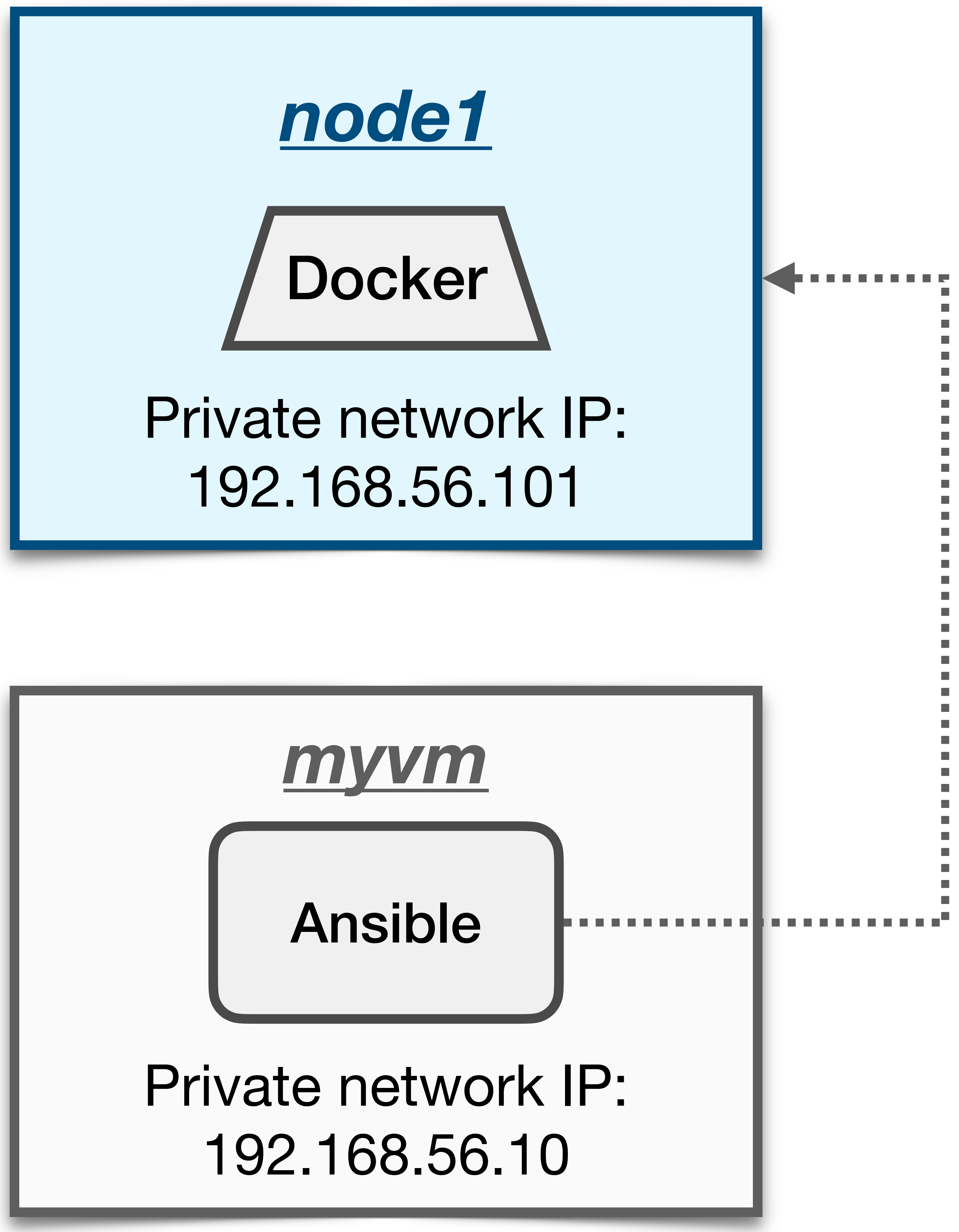
Useful Commands

- 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

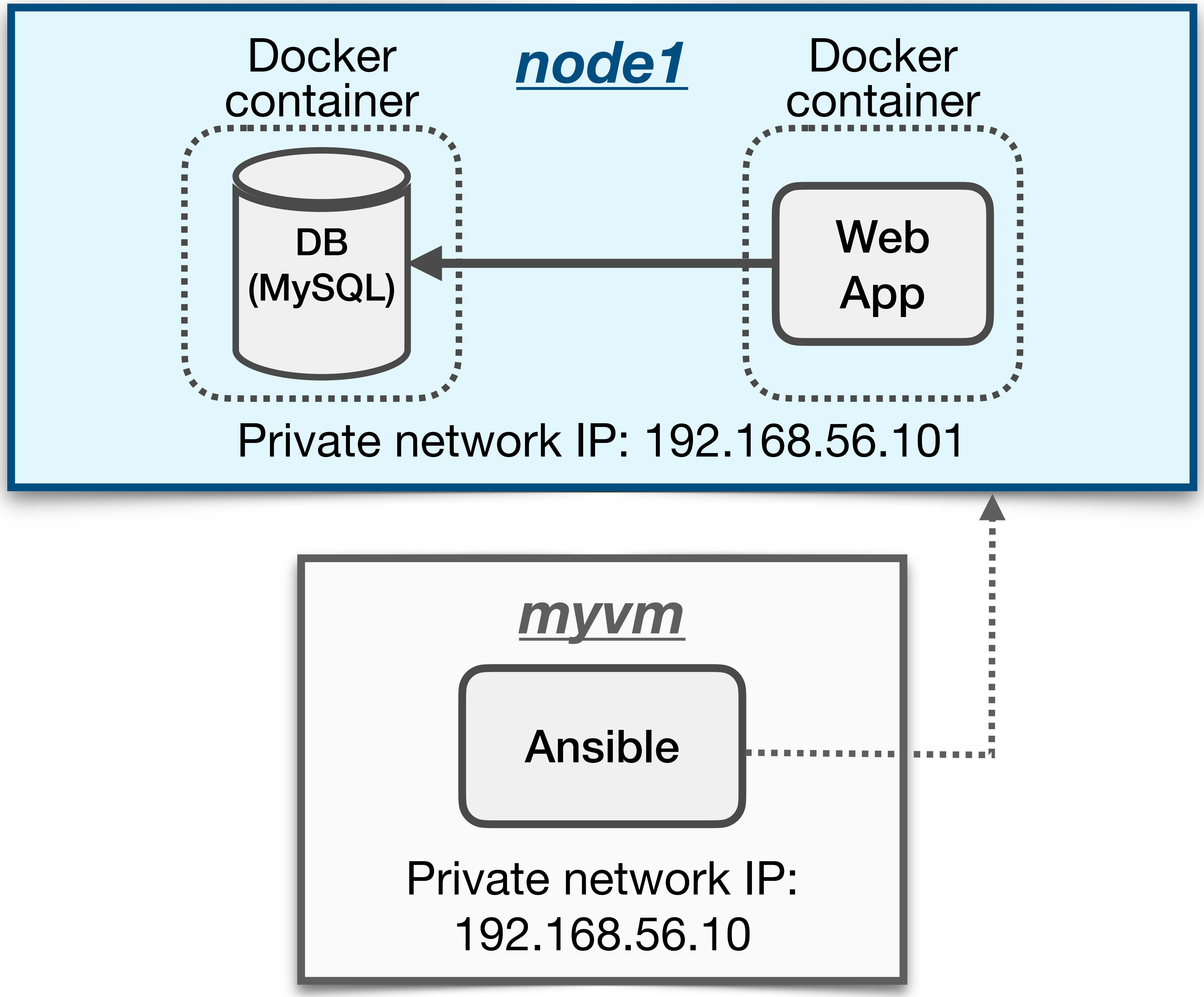
- Use Ansible to automate the installation and configuration of Docker on *node1.*


node1
Docker
Private network IP: 192.168.56.101
myvm
Ansible
Private network IP: 192.168.56.10

Part II - Goal

App Developer

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


node1
Docker container
Docker container
DB (MySQL)
Web App
Private network IP: 192.168.56.101
myvm
Ansible
Private network IP: 192.168.56.10

Part II - Material provided

This Guide is accompanied by two Ansible projects.
Please download, inspect them and follow your practical guide.

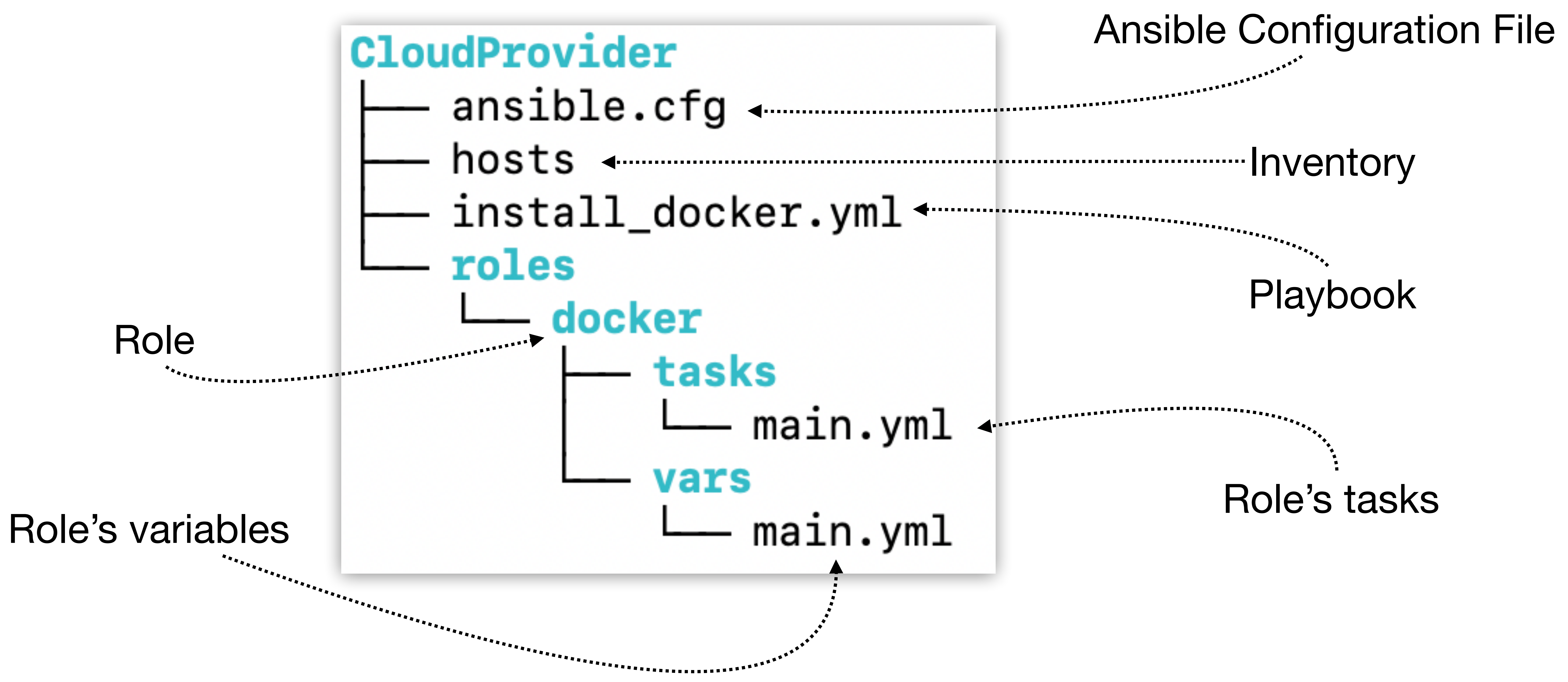
Cloud Provider

- ◉ 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

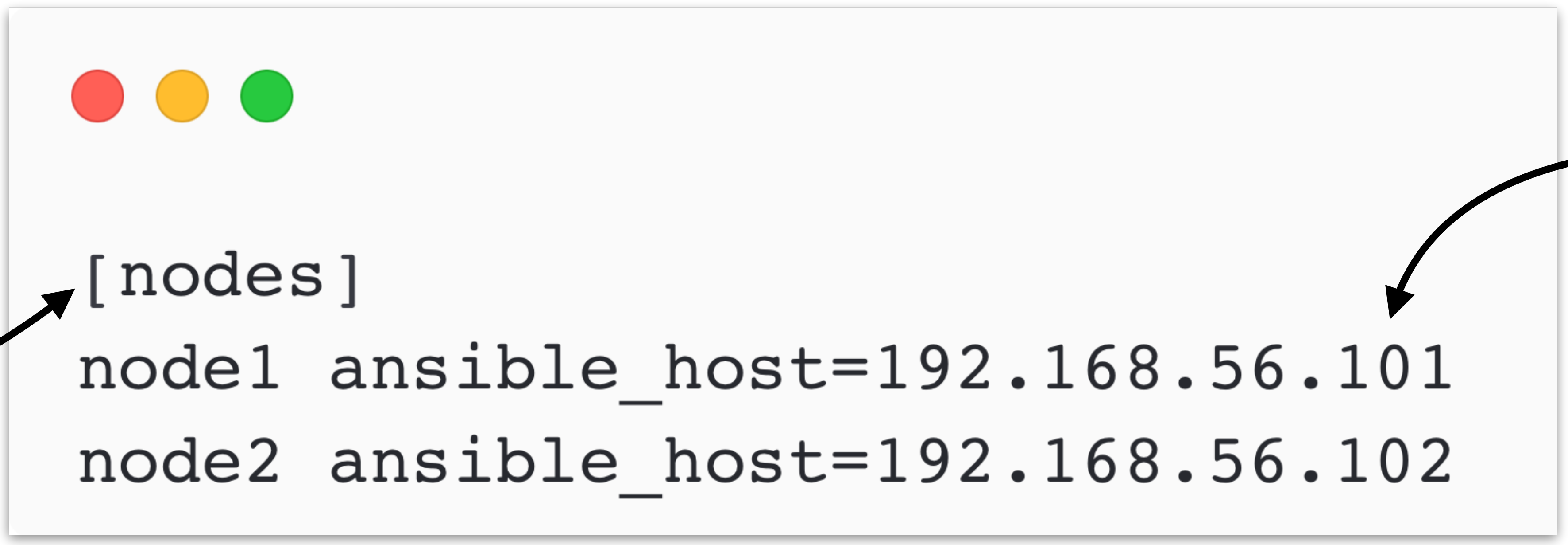
CloudProvider Project



```
CloudProvider
├── ansible.cfg
├── hosts
├── install_docker.yml
└── roles
    └── docker
        ├── tasks
        │   └── main.yml
        └── vars
            └── main.yml
```

CloudProvider Project

Inventory (hosts)

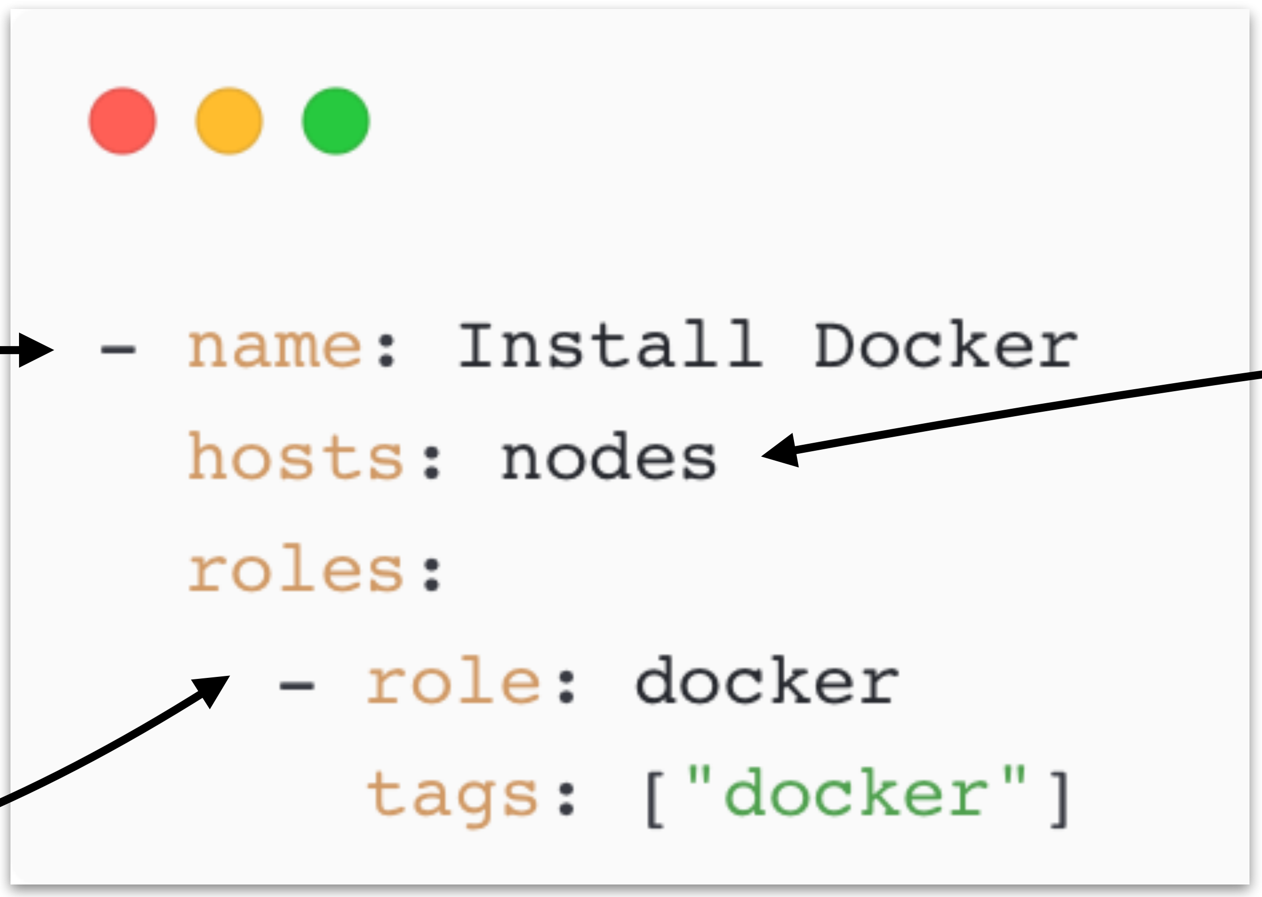


```
[nodes]
node1 ansible_host=192.168.56.101
node2 ansible_host=192.168.56.102
```

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

Task's name
The become directive allows for executing the task with sudo permissions
Ansible module
docker_dependencies_pkg is variable defined in the vars/main.yml file and used in the task.

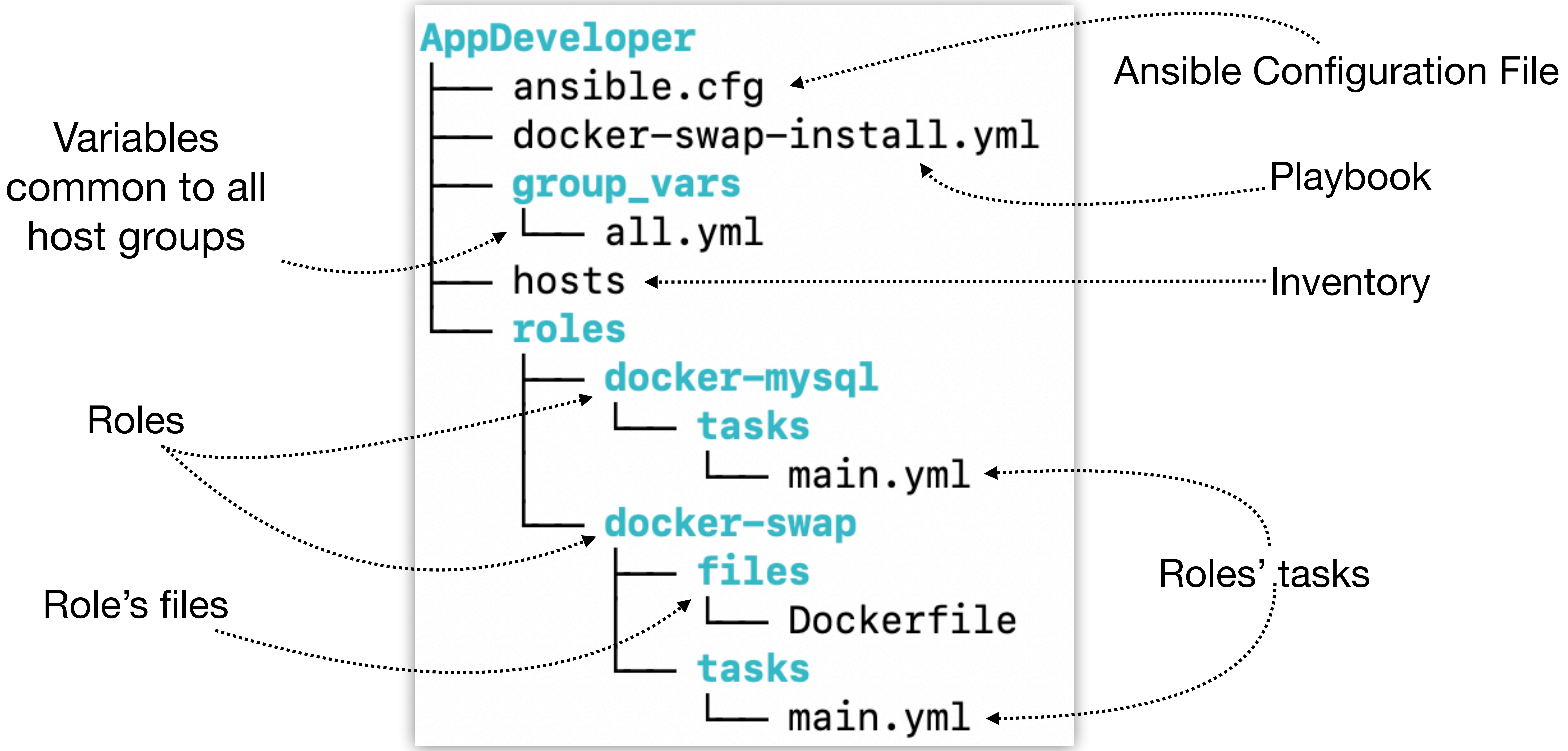
Role's tasks
(roles/docker/tasks/main.yml)

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

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

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

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

AppDeveloper Project



```
AppDeveloper
├── ansible.cfg
├── docker-swap-install.yml
├── group_vars
│   └── all.yml
├── hosts
└── roles
    ├── docker-mysql
    │   └── tasks
    │       └── main.yml
    └── docker-swap
        ├── files
        │   └── Dockerfile
        └── tasks
            └── main.yml
```

Ansible Configuration File
Playbook
Variables common to all host groups
Inventory
Roles
Roles' tasks
Role's files