

# School Of Information Sciences

(A constituent unit of Manipal Academy of Higher Education)



Project Report

On

## **ORCHISTRATION OF CONTAINERS USING ANSIBLE**

**Submitted by:**

(ME-VIR)

Harshitha K P – 181041008

Suman Banakar – 181041012

**Under the guidance of**

Mr Mohan Kumar J

School of Information Sciences,  
Lower Ground 2, Academic Block 5,  
MIT Campus, Manipal

# CONTENTS

LIST OF FIGURES	03
ABSTRACT	04
CHAPTER 1	
Introduction .....	05
Literature survey .....	06
Objectives .....	07
CHAPTER 2	
Software requirements .....	08
Installing Ansible .....	08
CHAPTER 3	
Block Diagram .....	09
Flow chart .....	09
Methodology .....	
RESULTS	
CONCLUSION	
REFERENCES	

## **LIST OF FIGURES**

1. Fig 3.1 Block diagram of orchestration of containers using ansible
2. Fig 3.2 Flow chart of orchestration of containers using ansible

# ABSTRACT

Ansible Container provides an Ansible-centric workflow for building, running, testing, and deploying containers.

Ansible Container enables to build container images and orchestrate them using only Ansible playbooks. Describe application in a single YAML file and, rather than using a Dockerfile, list Ansible roles that make up your container images. Ansible Container will take it from there. With Ansible Container, no need to build and configure containers differently than doing on traditional virtual machines or bare-metal systems. You can now apply the power of Ansible and re-use the existing Ansible content for our containerized ecosystem. Everything Ansible brings to orchestrating our infrastructure can now be applied to the image build process. But Ansible Container does not stop there. We can use Ansible Container to run the application, and Ansible playbook to automate the deployment.

The main objective of this project is to learn and generate ansible playbook that automates the deployment of containers on client machines.

***Keywords: Ansible, Playbook, Docker, Container***

## **CHAPTER 1**

### **Introduction**

Containers share the kernel of the underlying container host, however they don't need to run a full operating system. Rather, the software and dependencies are wrapped in a container that run on top of the container host. This allows a single container host (be it bare metal, virtual machine, or cloud offering) to host a substantially larger amount of services than the traditional server virtualization model. From the cost perspective, this allows further benefits that achieve greater economies of scale than what was previously possible.

Among the chief benefits of containers is the ability to develop in local environments on containers, and ship the container confidently from local dev through lower environments into production with exactly the same set of dependencies all safely wrapped in a container. Given the nature of containers and the multi-layered file system of Dockers, it's easy to rapidly deploy new images, pull down the most recent image, and commit new images to the container repository.

One of the key benefits to containerization applications is to develop on the same platform as production, which in this case is a container that has all dependencies wrapped in itself that can be shipped through various stages of deployment and various host operating systems with consistent results. Running containers locally allows for developers to work off the same consistent image both reducing the likelihood of incompatibilities from mismatched prod and dev environments.

Ansible is an automation tool that aims to ease tasks like configuration management, application deployment and intra-service orchestration. It has a built-in Docker module that integrates with Docker for container management.

Ansible is the way to automate Docker in your environment. Ansible enables you to operationalize your Docker container build and deployment process in ways that you're likely doing manually today, or not doing at all.

Ansible Playbooks to build containers, you gain the advantage of simple, repeatable, defined state of your containers that you can easily tarck.

Ansible can manage full enviornments. With Ansible you can manage not only the containers, but the enviornments around the containers, Docker instances still need to run on hosts, and those hosts need to be launched, configured, networked, and coordinated, whether they be local machines or full cloud infrastructures.

## **Literature survey**

### **1. Container-based virtual elastic clusters**

Usage of containers for Virtual elastic clouds allows better performance, fast deployment of additional working nodes, for enhanced elasticity.

The nodes of the cluster are hosted in containers running on bare-metal machines. The open- source tool Elastic Cluster for Docker (EC4Docker) is introduced, integrated with Docker Swarm to create auto-scaled virtual computer clusters of containers across distributed deployments

### **2. Orchestration of Containerized Microservices for IIoT using Docker**

IIOT relies on different devices working together, gathering and sharing data using multiple communication protocols. This heterogeneity becomes a hindrance in the development of architectures.

Docker simplifies management and enables distributed deployments and availability and fault-tolerance characteristics are ensured by distributing the application logic across different devices where a single microservice or even device failure can have no effect on system performance.

### **3. Orchestrating Docker Containers in the HPC Environment**

One of the requirements of grid computing(HPC) is to run a job transparently to the user on any resource they desire without requiring knowledge of the local software configuration. Based on our research and experimental results conducted, it is evident that Docker containers can facilitate this by abstracting the software environment of the local HPC resource without compromising performance. This improves Quality of Service

### **4. Container Orchestration for Scientific Workflows**

Data analysis platform designed to execute data-intensive scientific workflows. Recently they introduced Skypart, an extension to AWE/Shock, that uses Docker container technology to orchestrate and automate the deployment of individual workflow tasks onto the worker machines. Reduces complexity and offers an elegant solution to installation problems such as library version conflicts and offers a convenient and simple mechanism to reproduce scientific results.

## Objective

The main objective of this project is to create docker containers on Virtual machines using ansible playbook from host system, run the containers, and manage them.

## CHAPTER 2

### Software requirements

- Linux 16 and above
- Ansible
- VMWare Workstation
- Good internet connection for faster deployment

#### Steps for installing Ansible:

- `$ sudo apt-get update`
- `$sudo apt-get install python3-pip`
- `$sudo apt-get install software-properties-common`
- `$sudo -E apt-add-repository --yes --update ppa:ansible/ansible`
- `$sudo apt-get install ansible`
- `$ansible --version` ( To check installed ansible version)



## CHAPTER 3

### Methodology

#### Block Diagram:

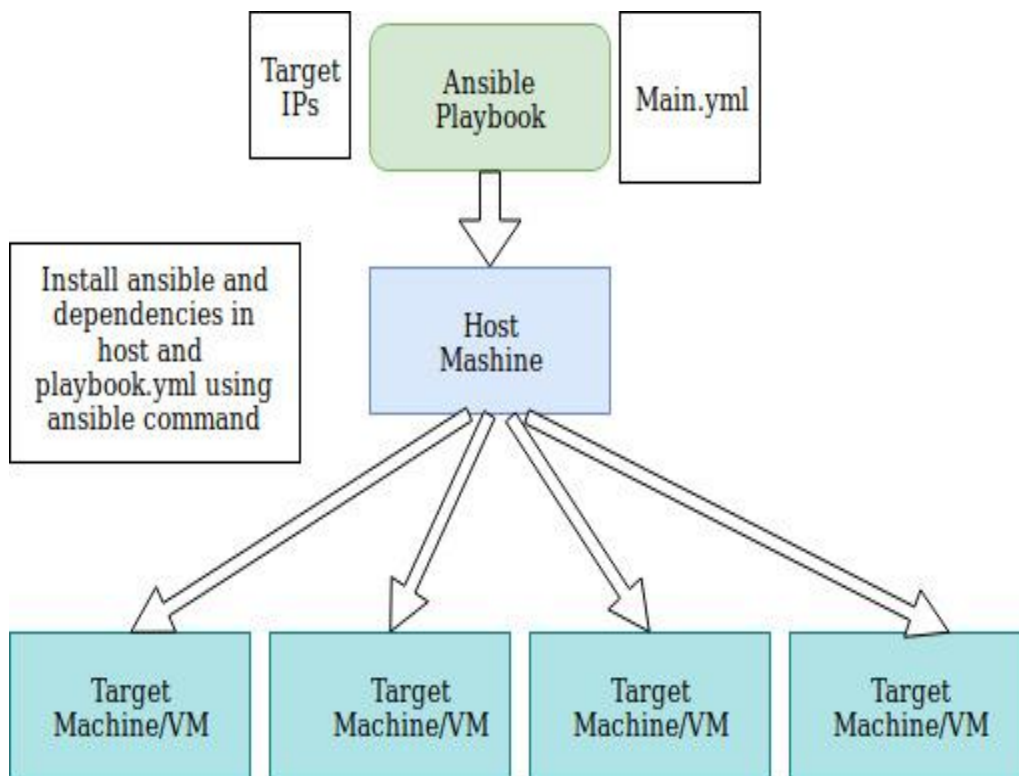


Fig 3.1 Block diagram of orchestration of containers using ansible

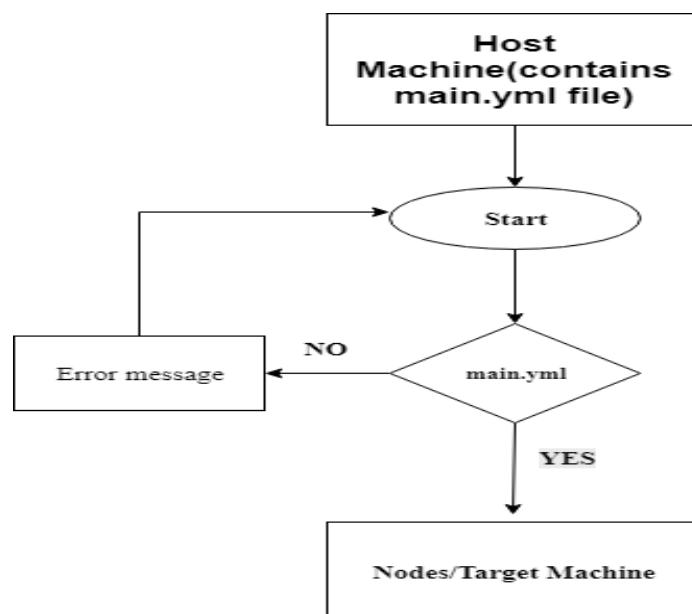


Fig 3.2 Flow chart of orchestration of containers using ansible

Step1: Start the host machine.

Step2: Run the main.yml file

Step3: Checks for the condition whether main.yml is successful,

If yes then containers are created on nodes/target machine,

If no then gives error message saying failed.

Step4: END.

### **Methodology:**

#### **Ansible on Linux**

##### **Step 1 — Build a docker image with Dockerfile**

First, create a directory on the workstation and a Dockerfile within the created directory.

```
$mkdir docker  
$cd docker  
$touch Dockerfile
```

Edit Dockerfile with nano and add the following content:

```
# Version: 0.0.1  
FROM Ubuntu:latest  
RUN apt-get update  
RUN apt-get install -y nginx  
RUN echo 'Default page. Nginx is in your container. ' \  
>/usr/share/nginx/html/index.html  
EXPOSE 80
```

##### **Step 2 — Edit inventory file**

Inventory file contains IP addresses or domain names where we want deploy container. Add this file in your project:

```
$touch ../hosts
```

Edit the hosts file with nano and add the following content within this file.

```
[webserver]
your_server_ip
```

Now, check if your local machine can communicate with the server via Ansible. Type the following command in terminal:

```
$ansible webserver -m ping -i ../hosts
```

### Step 3 — Edit Ansible playbook

First, change the working directory to the project root directory and create a new Ansible playbook. The directory layout should look like:

```
docker_project/
  main.yml
  hosts
  docker/
    Dockerfile
```

### Step 4 — Running the playbook

Run the playbook by using the following command on your workstation:

```
$ansible-playbook -i ~/hosts main.yml --ask-become-pass --ask-sudo-pass --ask-pass
```

### Modules:

Ansible works by connecting to your nodes and pushing out small programs, called “Ansible Modules” to them. These programs are written to be resource models of the desired state of the system. Ansible then executes these modules (over SSH by default), and removes them when finished.

Your library of modules can reside on any machine, and there are no servers, daemons, or databases required. Typically, you’ll work with your favourite terminal program, a text editor, and probably a version control system to keep track of changes to your content.

### **Inventory:**

By default, Ansible represents what machines it manages using a very simple INI file that puts all of your managed machines in groups of your own choosing. To add new machines, there is no additional SSL signing server involved, so there's never any hassle deciding why a particular machine didn't get linked up due to obscure NTP or DNS issues. If there's another source of truth in your infrastructure, Ansible can also plugin to that, such as drawing inventory, group, and variable information from sources like EC2, Rackspace, OpenStack, and more. Once inventory hosts are listed, variables can be assigned to them in simple text files (in a subdirectory called 'group\_vars/' or 'host\_vars/' or directly in the inventory file. Or, as already mentioned, use a dynamic inventory to pull your inventory from data sources like EC2, Rackspace, or OpenStack.

### **Playbooks and Roles:**

Ansible performs automation and orchestration of computing environments using play books. Playbooks are a YAML definition of automation tasks that describe how a particular piece of automation should be done.

Playbooks are written in a descriptive manner that clearly states what each individual component of your infrastructure needs to do, but still allows components to react to discovered information and to operate in concert with each other. An example of a simple playbook is shown in Figure 3.1.

Ansible playbooks consist of series of plays that define automation across a set of hosts, known as the inventory. Each play consists of multiple tasks that can target one, many, or all of the hosts in the inventory. Each task is a call to an Ansible module: a small piece of code for doing a specific task. These tasks can be simple, such as placing a configuration file on a target machine or installing a software package. They can be complex, such as provisioning and mapping storage for an entire NetApp E - Series data centre.

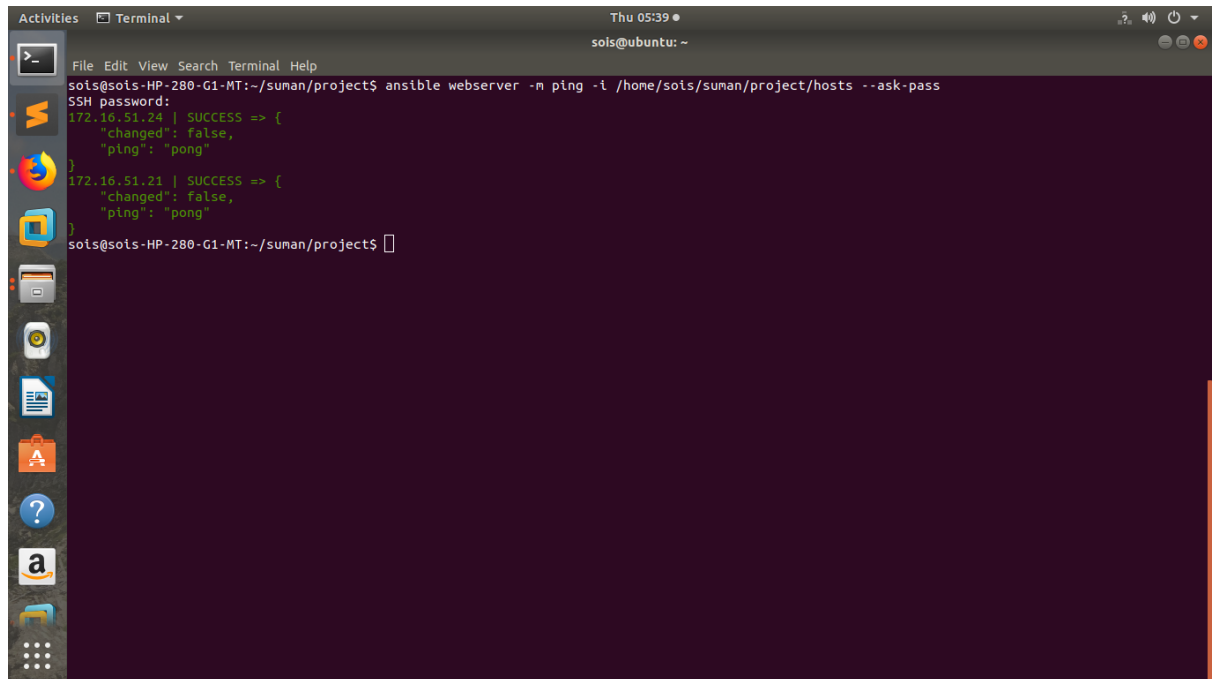
Ansible modules are written to allow for easy configuration of a desired state. They check that the task that is specified actually needs to be done before executing it. For example, an Ansible module is defined to create a Dynamic Disk Pool (DDP), but configuration is only done if the DDP is not already created.

This desired state configuration, sometimes referred to as idempotency, makes sure that configuration can be applied repeatedly without side effects and that configuration runs quickly and efficiently when it has already been applied.

Ansible also supports encapsulating playbook tasks into reusable units called roles. Ansible roles can be used to easily apply common configurations in different scenarios, such as having a common DDP configuration role that may be used in development, test, or production automation.

## RESULTS

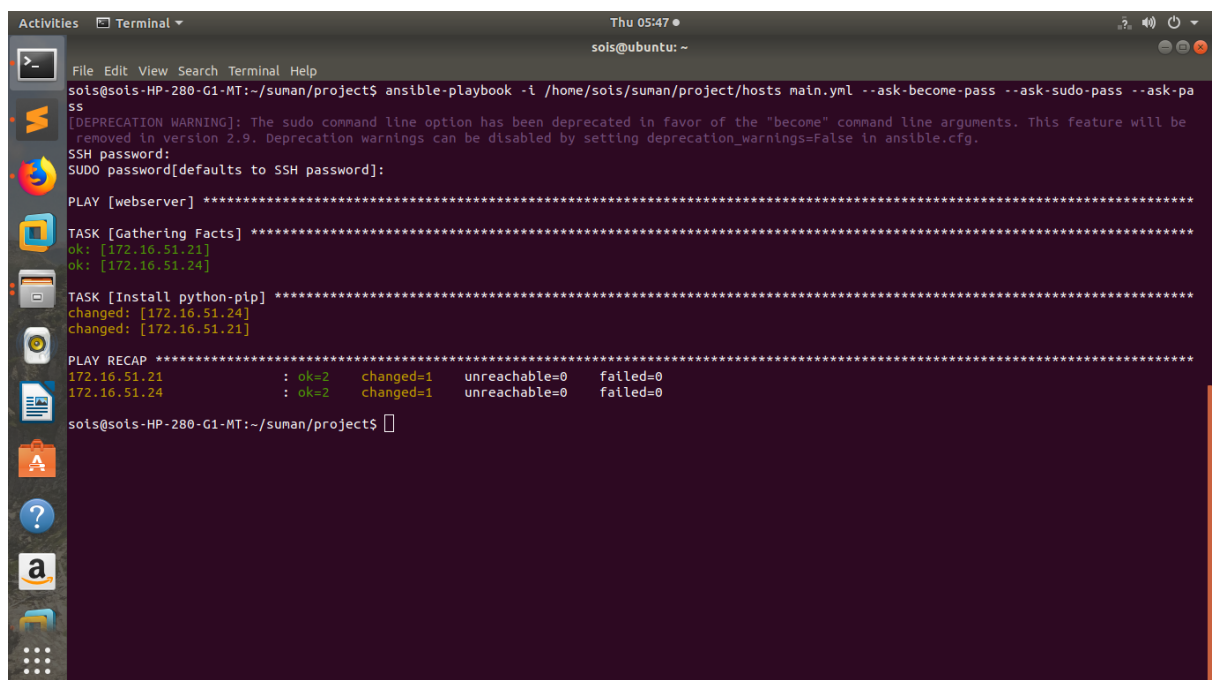
### 1. To check SSH request is working



```

sois@sois-HP-280-G1-MT:~/suman/project$ ansible webserver -m ping -i /home/sois/suman/project/hosts --ask-pass
SSH password:
172.16.51.24 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
172.16.51.21 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
sois@sois-HP-280-G1-MT:~/suman/project$
  
```

### 2. Install python pip on to node



```

sois@sois-HP-280-G1-MT:~/suman/project$ ansible-playbook -i /home/sois/suman/project/hosts main.yml --ask-become-pass --ask-sudo-pass --ask-pass
[DEPRECATION WARNING]: The sudo command line option has been deprecated in favor of the "become" command line arguments. This feature will be
removed in version 2.9. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
SSH password:
SUDO password(defaults to SSH password):

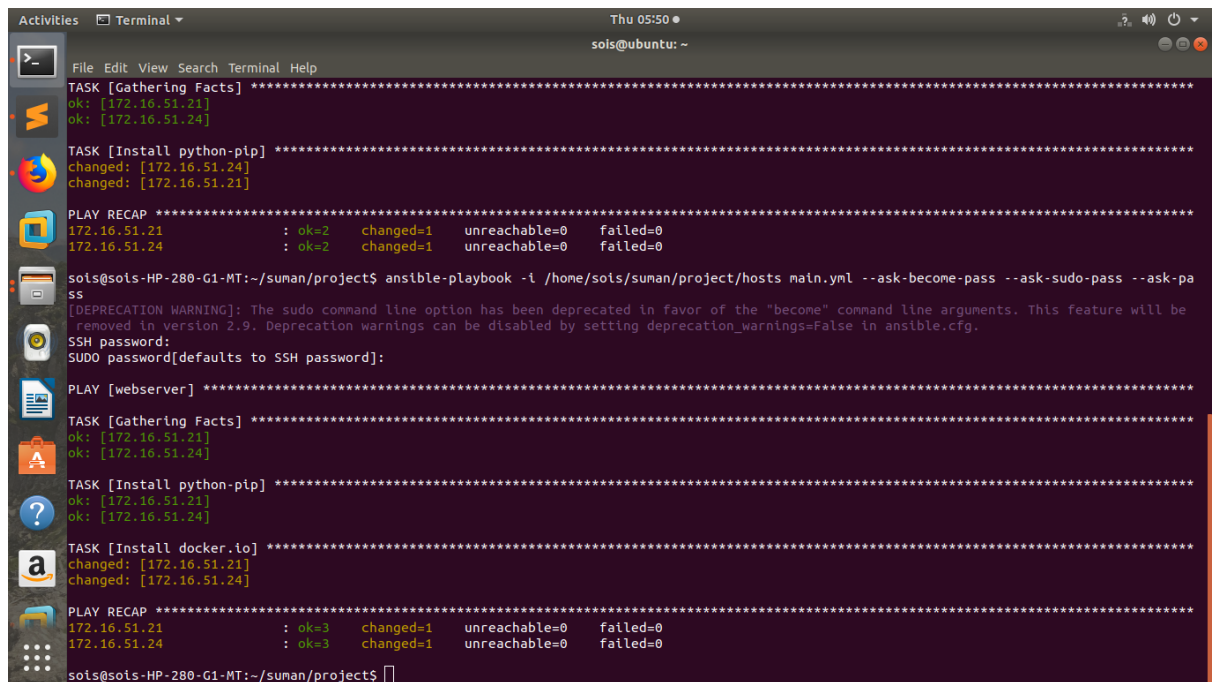
PLAY [webserver] *****

TASK [Gathering Facts] *****
ok: [172.16.51.21]
ok: [172.16.51.24]

TASK [Install python-pip] *****
changed: [172.16.51.24]
changed: [172.16.51.21]

PLAY RECAP *****
172.16.51.21      : ok=2    changed=1    unreachable=0    failed=0
172.16.51.24      : ok=2    changed=1    unreachable=0    failed=0
sois@sois-HP-280-G1-MT:~/suman/project$
  
```

### 3. Install Docker.io



The terminal window shows the execution of an Ansible playbook. The first play, 'webserver', includes tasks for gathering facts, installing python-pip, and installing docker.io. The second play, 'docker', includes a task for installing docker-py. The output shows that docker.io was installed on both hosts (172.16.51.21 and 172.16.51.24) and docker-py was installed on 172.16.51.24.

```
soils@soils-HP-280-G1-MT:~/suman/project$ ansible-playbook -i /home/soils/suman/project/hosts main.yml --ask-become-pass --ask-sudo-pass --ask-pass
[DEPRECATION WARNING]: The sudo command line option has been deprecated in favor of the "become" command line arguments. This feature will be
removed in version 2.9. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
SSH password:
SUDO password[defaults to SSH password]:

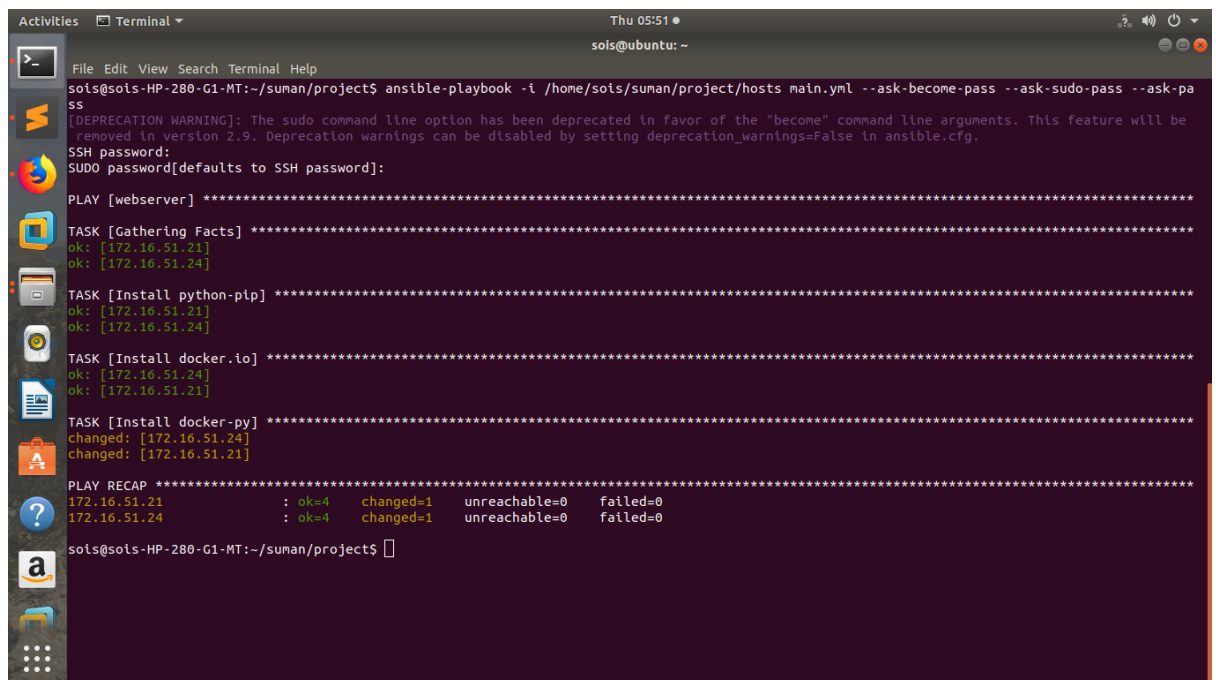
PLAY [webserver] *****
TASK [Gathering Facts] *****
ok: [172.16.51.21]
ok: [172.16.51.24]
TASK [Install python-pip] *****
changed: [172.16.51.24]
changed: [172.16.51.21]
PLAY RECAP *****
172.16.51.21 : ok=2 changed=1 unreachable=0 failed=0
172.16.51.24 : ok=2 changed=1 unreachable=0 failed=0

soils@soils-HP-280-G1-MT:~/suman/project$ ansible-playbook -i /home/soils/suman/project/hosts main.yml --ask-become-pass --ask-sudo-pass --ask-pass
[DEPRECATION WARNING]: The sudo command line option has been deprecated in favor of the "become" command line arguments. This feature will be
removed in version 2.9. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
SSH password:
SUDO password[defaults to SSH password]:

PLAY [webserver] *****
TASK [Gathering Facts] *****
ok: [172.16.51.21]
ok: [172.16.51.24]
TASK [Install python-pip] *****
ok: [172.16.51.21]
ok: [172.16.51.24]
TASK [Install docker.io] *****
changed: [172.16.51.21]
changed: [172.16.51.24]
PLAY RECAP *****
172.16.51.21 : ok=3 changed=1 unreachable=0 failed=0
172.16.51.24 : ok=3 changed=1 unreachable=0 failed=0

soils@soils-HP-280-G1-MT:~/suman/project$
```

### 4. Install Docker py



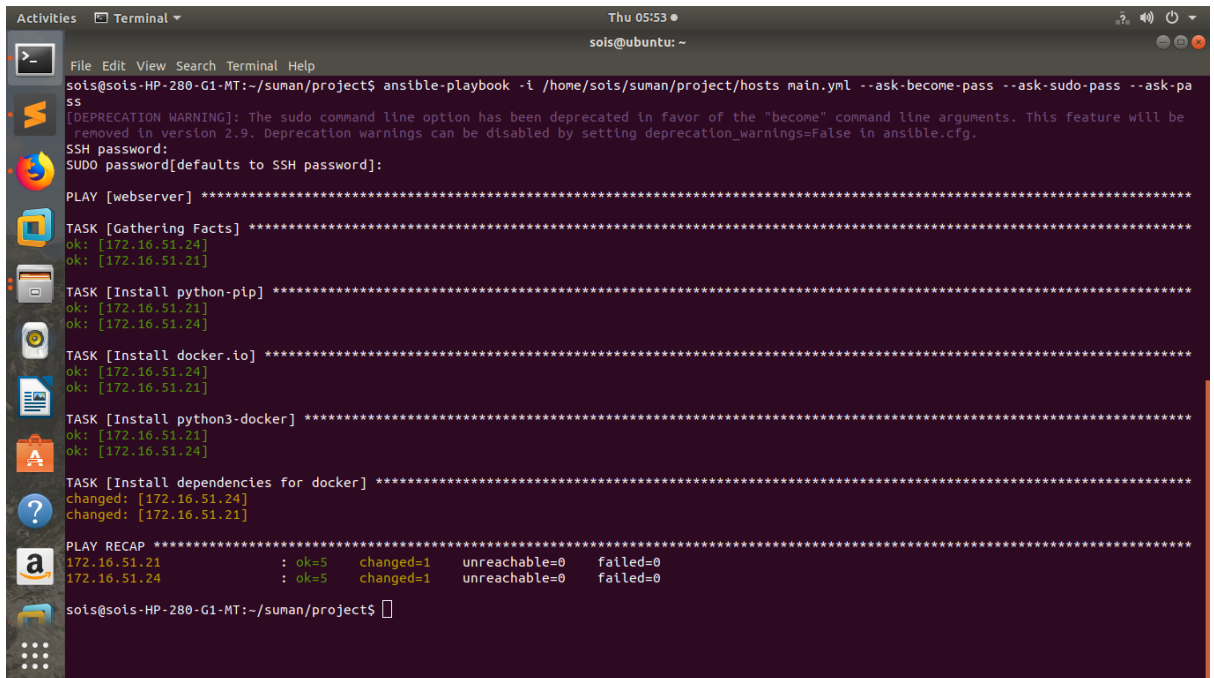
The terminal window shows the execution of an Ansible playbook. The first play, 'webserver', includes tasks for gathering facts, installing python-pip, and installing docker.io. The second play, 'docker', includes a task for installing docker-py. The output shows that docker-py was installed on both hosts (172.16.51.21 and 172.16.51.24).

```
soils@soils-HP-280-G1-MT:~/suman/project$ ansible-playbook -i /home/soils/suman/project/hosts main.yml --ask-become-pass --ask-sudo-pass --ask-pass
[DEPRECATION WARNING]: The sudo command line option has been deprecated in favor of the "become" command line arguments. This feature will be
removed in version 2.9. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
SSH password:
SUDO password[defaults to SSH password]:

PLAY [webserver] *****
TASK [Gathering Facts] *****
ok: [172.16.51.21]
ok: [172.16.51.24]
TASK [Install python-pip] *****
ok: [172.16.51.21]
ok: [172.16.51.24]
TASK [Install docker.io] *****
ok: [172.16.51.21]
ok: [172.16.51.24]
TASK [Install docker-py] *****
changed: [172.16.51.24]
changed: [172.16.51.21]
PLAY RECAP *****
172.16.51.21 : ok=4 changed=1 unreachable=0 failed=0
172.16.51.24 : ok=4 changed=1 unreachable=0 failed=0

soils@soils-HP-280-G1-MT:~/suman/project$
```

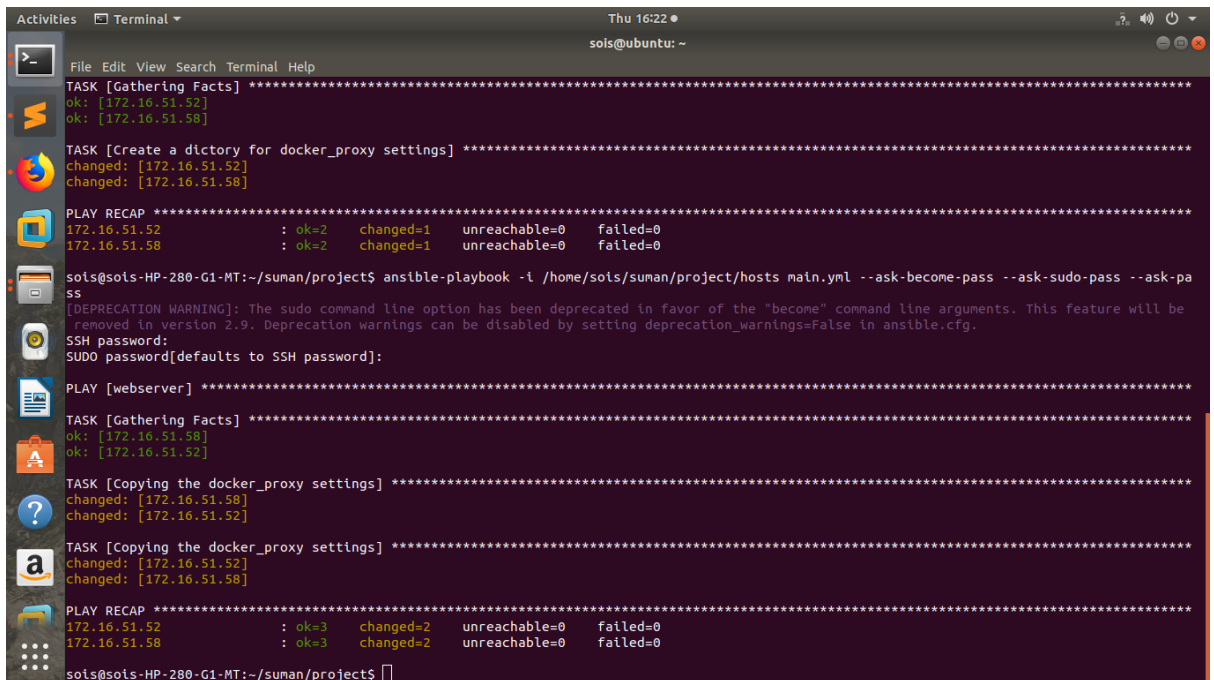
### 5. Install dependencies of docker:



```
soils@soils-HP-280-G1-MT:~/sunan/project$ ansible-playbook -i /home/soils/sunan/project/hosts main.yml --ask-become-pass --ask-sudo-pass --ask-pass
[DEPRECATION WARNING]: The sudo command line option has been deprecated in favor of the "become" command line arguments. This feature will be
removed in version 2.9. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
SSH password:
SUDO password[defaults to SSH password]:

PLAY [webserver] *****
TASK [Gathering Facts] *****
ok: [172.16.51.24]
ok: [172.16.51.21]
TASK [Install python-pip] *****
ok: [172.16.51.21]
ok: [172.16.51.24]
TASK [Install docker.io] *****
ok: [172.16.51.24]
ok: [172.16.51.21]
TASK [Install python3-docker] *****
ok: [172.16.51.21]
ok: [172.16.51.24]
TASK [Install dependencies for docker] *****
changed: [172.16.51.24]
changed: [172.16.51.21]
PLAY RECAP *****
172.16.51.21 : ok=5 changed=1 unreachable=0 failed=0
172.16.51.24 : ok=5 changed=1 unreachable=0 failed=0
soils@soils-HP-280-G1-MT:~/sunan/project$
```

### 6. Create directory for proxy settings and copying proxy files:



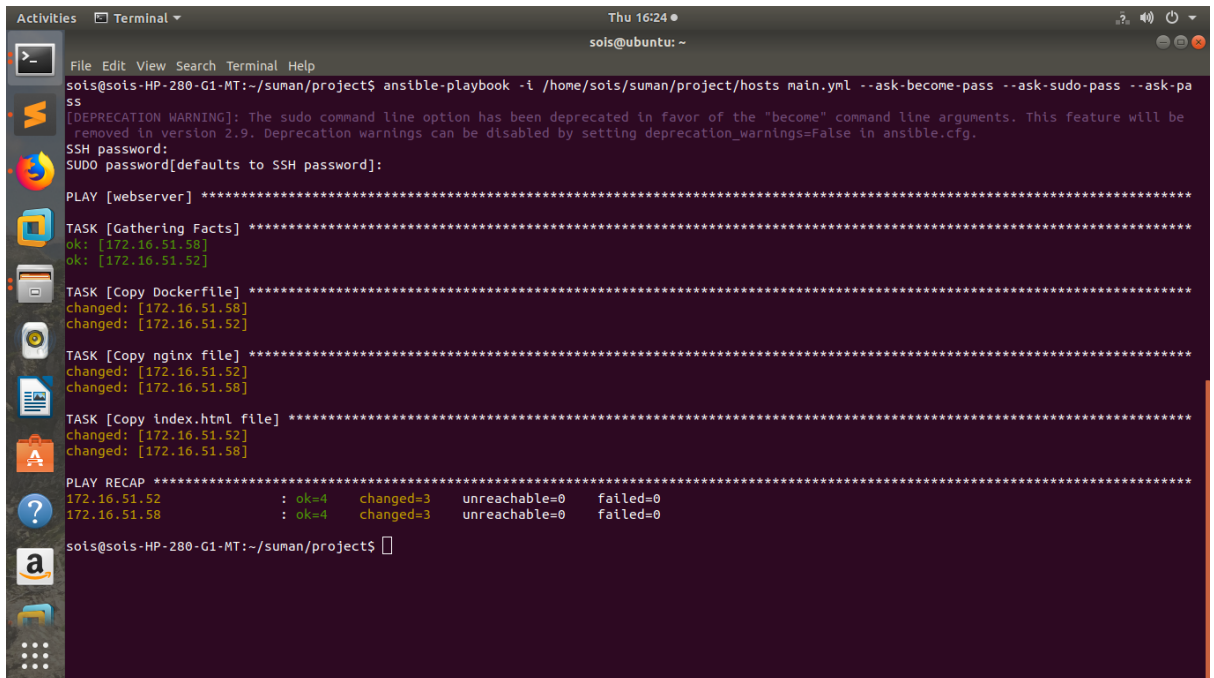
```
soils@soils-HP-280-G1-MT:~/sunan/project$ ansible-playbook -i /home/soils/sunan/project/hosts main.yml --ask-become-pass --ask-sudo-pass --ask-pass
[DEPRECATION WARNING]: The sudo command line option has been deprecated in favor of the "become" command line arguments. This feature will be
removed in version 2.9. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
SSH password:
SUDO password[defaults to SSH password]:

PLAY [webserver] *****
TASK [Gathering Facts] *****
ok: [172.16.51.52]
ok: [172.16.51.58]
TASK [Create a dictory for docker_proxy settings] *****
changed: [172.16.51.52]
changed: [172.16.51.58]
PLAY RECAP *****
172.16.51.52 : ok=2 changed=1 unreachable=0 failed=0
172.16.51.58 : ok=2 changed=1 unreachable=0 failed=0
soils@soils-HP-280-G1-MT:~/sunan/project$ ansible-playbook -i /home/soils/sunan/project/hosts main.yml --ask-become-pass --ask-sudo-pass --ask-pass
[DEPRECATION WARNING]: The sudo command line option has been deprecated in favor of the "become" command line arguments. This feature will be
removed in version 2.9. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
SSH password:
SUDO password[defaults to SSH password]:

PLAY [webserver] *****
TASK [Gathering Facts] *****
ok: [172.16.51.58]
ok: [172.16.51.52]
TASK [Copying the docker_proxy settings] *****
changed: [172.16.51.58]
changed: [172.16.51.52]
TASK [Copying the docker_proxy settings] *****
changed: [172.16.51.52]
changed: [172.16.51.58]
PLAY RECAP *****
172.16.51.52 : ok=3 changed=2 unreachable=0 failed=0
172.16.51.58 : ok=3 changed=2 unreachable=0 failed=0
soils@soils-HP-280-G1-MT:~/sunan/project$
```



### 7. Copying Nginx Dockerfile



```
sois@sois-HP-280-G1-MT:~/suman/project$ ansible-playbook -i /home/sois/suman/project/hosts main.yml --ask-become-pass --ask-sudo-pass --ask-pass
[DEPRECATION WARNING]: The sudo command line option has been deprecated in favor of the "become" command line arguments. This feature will be
removed in version 2.9. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
SSH password:
SUDO password[defaults to SSH password]:

PLAY [webserver] *****

TASK [Gathering Facts] *****
ok: [172.16.51.58]
ok: [172.16.51.52]

TASK [Copy Dockerfile] *****
changed: [172.16.51.58]
changed: [172.16.51.52]

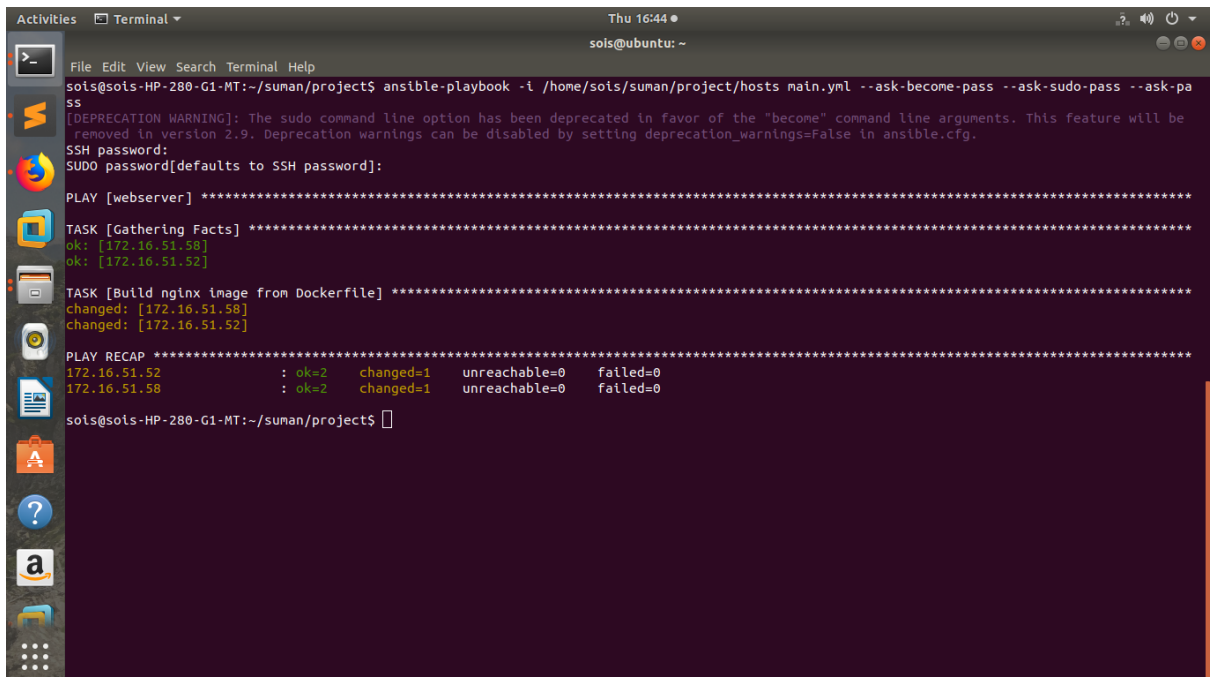
TASK [Copy nginx file] *****
changed: [172.16.51.52]
changed: [172.16.51.58]

TASK [Copy index.html file] *****
changed: [172.16.51.52]
changed: [172.16.51.58]

PLAY RECAP *****
172.16.51.52      : ok=4    changed=3    unreachable=0    failed=0
172.16.51.58      : ok=4    changed=3    unreachable=0    failed=0

sois@sois-HP-280-G1-MT:~/suman/project$
```

### 8. Build Nginx image from Dockerfile



```
sois@sois-HP-280-G1-MT:~/suman/project$ ansible-playbook -i /home/sois/suman/project/hosts main.yml --ask-become-pass --ask-sudo-pass --ask-pass
[DEPRECATION WARNING]: The sudo command line option has been deprecated in favor of the "become" command line arguments. This feature will be
removed in version 2.9. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
SSH password:
SUDO password[defaults to SSH password]:

PLAY [webserver] *****

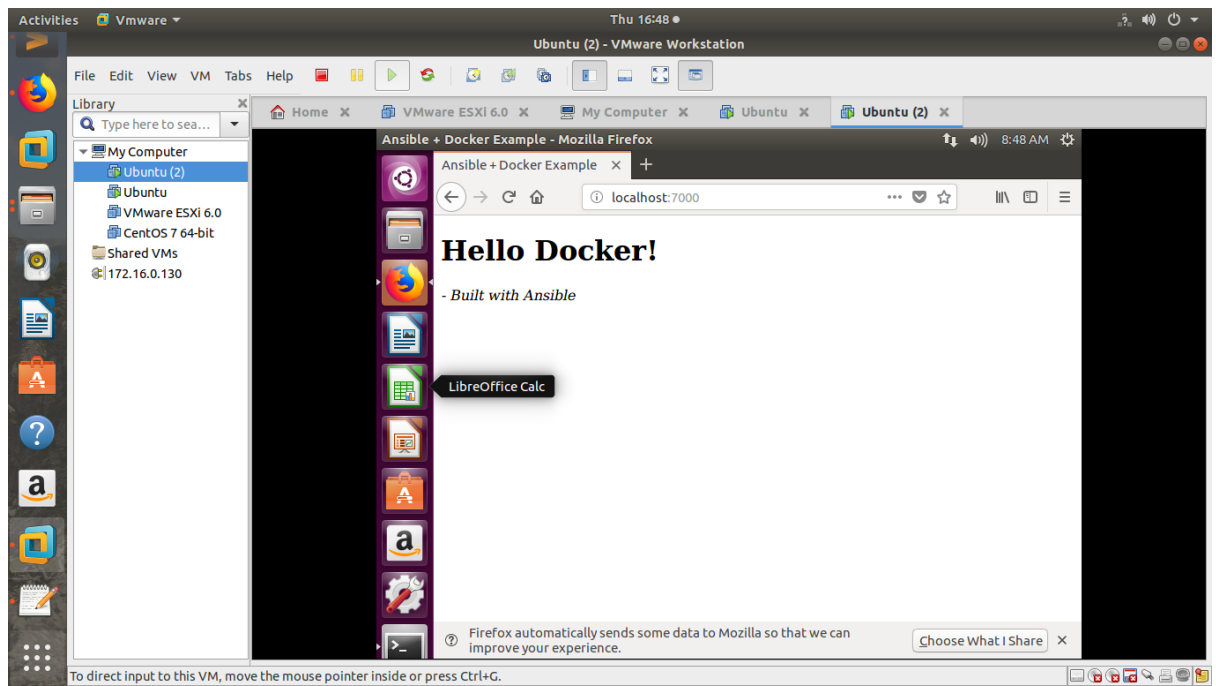
TASK [Gathering Facts] *****
ok: [172.16.51.58]
ok: [172.16.51.52]

TASK [Build nginx image from Dockerfile] *****
changed: [172.16.51.58]
changed: [172.16.51.52]

PLAY RECAP *****
172.16.51.52      : ok=2    changed=1    unreachable=0    failed=0
172.16.51.58      : ok=2    changed=1    unreachable=0    failed=0

sois@sois-HP-280-G1-MT:~/suman/project$
```

### 9. Running an Nginx



### CONCLUSION

Ansible automates Docker in our environment. Ansible enables to operationalize Docker container build and deployment process in ways that we likely do it manually.

By using ansible Playbooks which are portable. We can build a container with a pure Dockerfile, Building a container with an Ansible Playbook, we can then reproduce your environment in Docker on Virtual machines or on bare metal. We can build our containers up using Ansible Roles, and different container roles can be reused across many environments.

## REFERENCES

1. [www.docker.com](http://www.docker.com)
2. [www.Hub.docker.com](http://www.Hub.docker.com)
3. <https://www.codementor.io/mamytianarakotomalala/how-to-deploy-docker-container-with-ansible-on-debian-8-mavm48kw0>
4. [https://docs.ansible.com/ansible/2.5/modules/docker\\_module.html](https://docs.ansible.com/ansible/2.5/modules/docker_module.html)
5. Container-based virtual elastic clusters Carlos de Alfonso \*, Amanda Calatrava, Germán Moltó *Instituto de Instrumentación para Imagen Molecular (I3M), Centro mixto CSIC - Universidad Politécnica de Valencia - CIEMAT, Camino de Vera s/n, 46022, Valencia, Spain*
6. Orchestration of Containerized Microservices for IIoT using Docker  
João Rufino\*, Muhammad Alam\*, Joaquim Ferreira\*,†, Abdur Rehman‡, Kim Fung Tsang\*\*\* Instituto de Telecomunicações, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal ‡ ESTGA - Universidade de Aveiro, 3754-909 A´gueda, Portugal †Department of Internetworking, Faculty of Engineering , Dalhousie University, Halifax, NS, Canada.\*\*City University of Hong Kong – HK. {joao.rufino, alam, jjcf}@ua.pt, abdur.rehman@dal.ca, [ee330015@cityu.edu.hk](mailto:ee330015@cityu.edu.hk)
7. Joshua Higgins(B), Violeta Holmes, and Colin Venters The University of Huddersfield, Queensgate, Huddersfield, UK {joshua.higgins,v.holmes,c.venters}@hud.ac.uk[3]
8. 2015 IEEE International Conference on Cloud Engineering[4]