

Mesos-Marathon

Setup a single mesos-master, marathon and mesos-slave. Try launching a docker app via marathon on the mesos-slave."

Ans:

1. Install java 8 using the command "sudo apt-get install -y openjdk-8-jdk". This serves as a dependency of Mesos and Marathon.
2. Add the mesos repository in /etc/apt/sources.list.d /mesosphere.list deb <http://repos.mesosphere.com/ubuntu> xenial main
3. Install mesos by running the following command "sudo apt-get install mesos" 4. Install marathon by running the following command "sudo apt-get install marathon"
5. Install docker engine by running the following command "sudo apt-get install docker.io"
6. To check the status of docker engine run the "sudo systemctl status docker" command. On the successful installation of Docker Engine, it will show the status of docker engine as Active.
7. Now start the Mesos-Master service by executing the following command service mesos-master start
8. Configure mesos and docker containerizer by creating containerizer file inside /etc/mesos-slave/containerizers
docker,mesos
9. Now start the Mesos-Slave service by executing the following command service mesos-slave start
10. Go to sudo vi /etc/default/marathon file and add these lines

```
##### Environment Configuration #####
#####

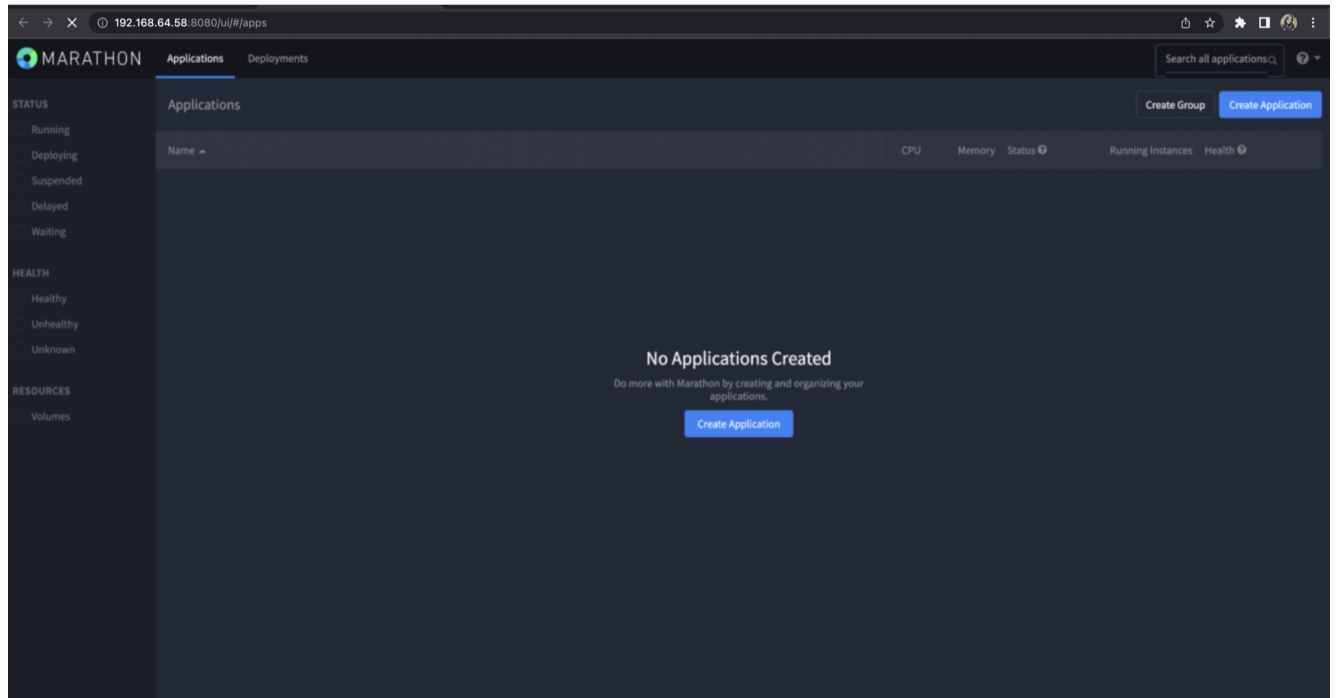
# Setting --mesos_user
# -----
MARATHON_MESOS_USER=root
MARATHON_MASTER=zk://127.0.0.1:2181/mesos
MARATHON_ZK=zk://127.0.0.1/marathon

# Setting JAVA_OPTS
# -----
# JAVA_OPTS="-Dpidfile.path=/var/run/marathon/marathon.pid"
```

11. After following the above steps, one can access Mesos UI Dashboard at port 5050 and Marathon UI Dashboard at port 8080.

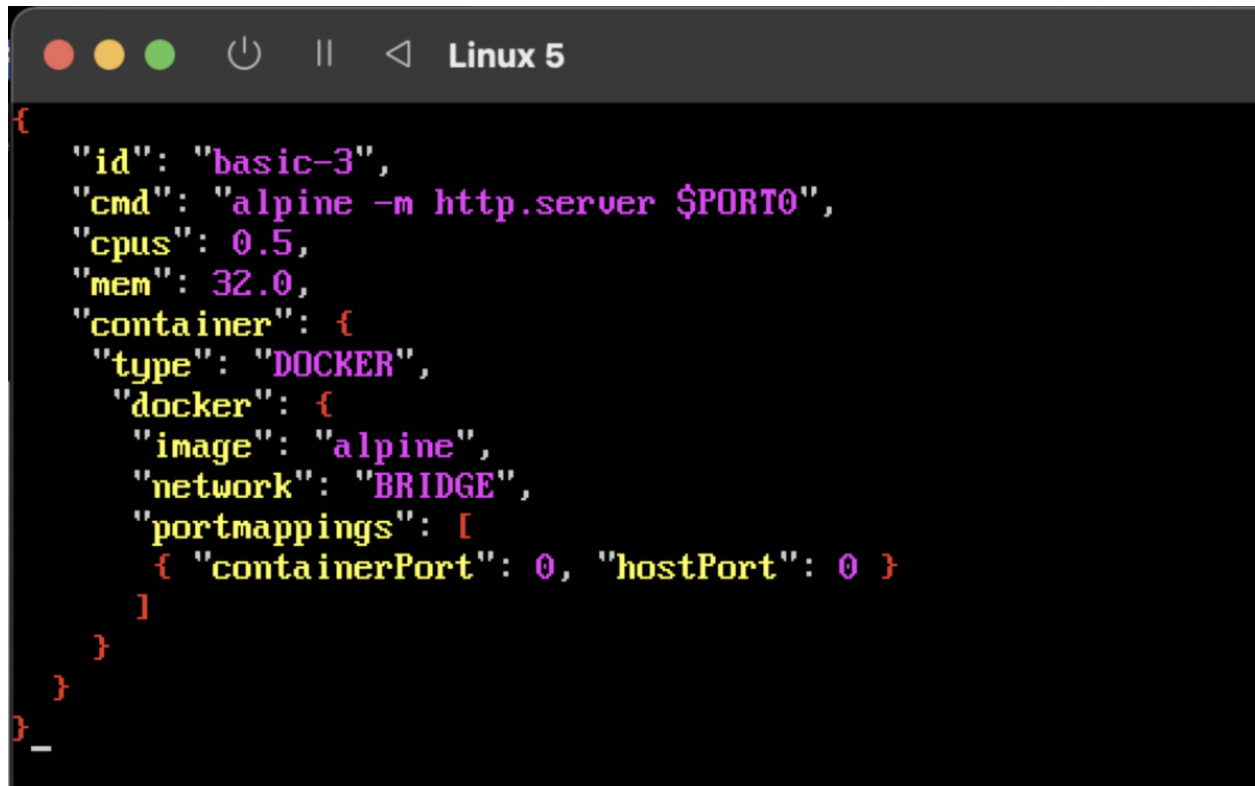
The screenshot shows the Mesos UI Dashboard in a web browser. The address bar indicates the URL is 192.168.64.58:5050/#. The dashboard has a blue header with the Mesos logo and navigation links: Frameworks, Agents, Roles, Offers, and Maintenance. Below the header, there's a 'Master' tab selected. The main content area is divided into several sections:

- Cluster Information:** Cluster: (Unnamed), Leader: (Unknown), Version: (Unknown), Built: by, Started: (Unknown), Elected: (Unknown).
- Agents:** A list of agents with status: Activated, Deactivated, Unreachable.
- Tasks:** A list of tasks with status: Staging (0), Starting (0), Running (0), Unreachable (0), Killing (0), Finished (0), Killed (0), Failed (0), Lost (0).
- Resources:** A table showing resource usage for CPUs, GPUs, Mem, and Disk. The table has rows for Total, Allocated, Offered, and Idle.
- Active Tasks:** A table with columns: Framework ID, Task ID, Task Name, Role, State, Health, Started (dropdown), and Host.
- Unreachable Tasks:** A table with columns: Framework ID, Task ID, Task Name, Role, Started (dropdown), and Agent ID.
- Completed Tasks:** A table with columns: Framework ID, Task ID, Task Name, Role, State, Started (dropdown), Stopped, and Host.



12. To verify all the components such as mesos-master, mesos-slave, zookeeper and marathon are active and running, run “sudo systemctl status <component>” command.

13. Create an app.json file.

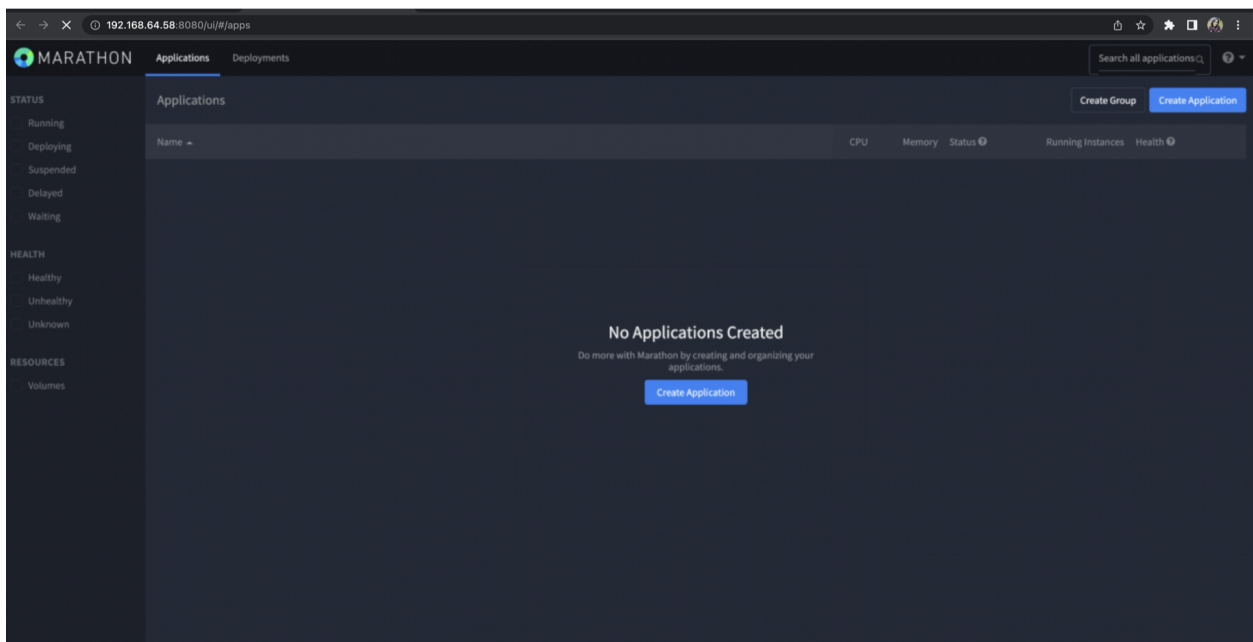
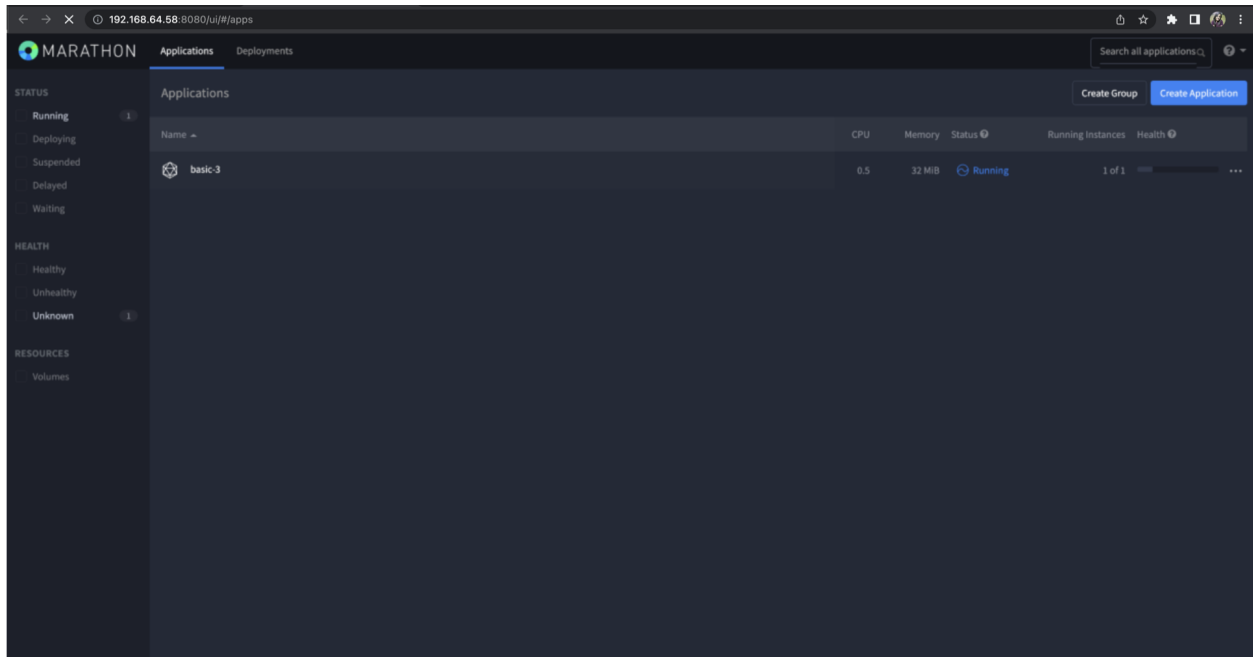
A terminal window titled "Linux 5" with standard window controls (red, yellow, green buttons and a power icon). The terminal displays a JSON configuration for a Docker container. The JSON is color-coded: keys are yellow, string values are purple, and numeric values are green. The configuration includes an ID, command, CPU and memory limits, container type, Docker-specific settings like image, network, and port mappings.

```
{
  "id": "basic-3",
  "cmd": "alpine -m http.server $PORT0",
  "cpus": 0.5,
  "mem": 32.0,
  "container": {
    "type": "DOCKER",
    "docker": {
      "image": "alpine",
      "network": "BRIDGE",
      "portmappings": [
        { "containerPort": 0, "hostPort": 0 }
      ]
    }
  }
}
```

14. Create an application using curl command

```
curl -X POST http://127.0.0.1:8080/v2/apps -d @app.json -H "Content-type: application/json"
```

16. On opening the marathon UI dashboard on the browser, we can view the application is running.



17. Run “docker ps”

```
Marathon@Mesos:~$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
16edd0b24d47   alpine        "/bin/sh -c 'alpine ..."  12 seconds ago Up 2 seconds  0.0.0.0:31211
```