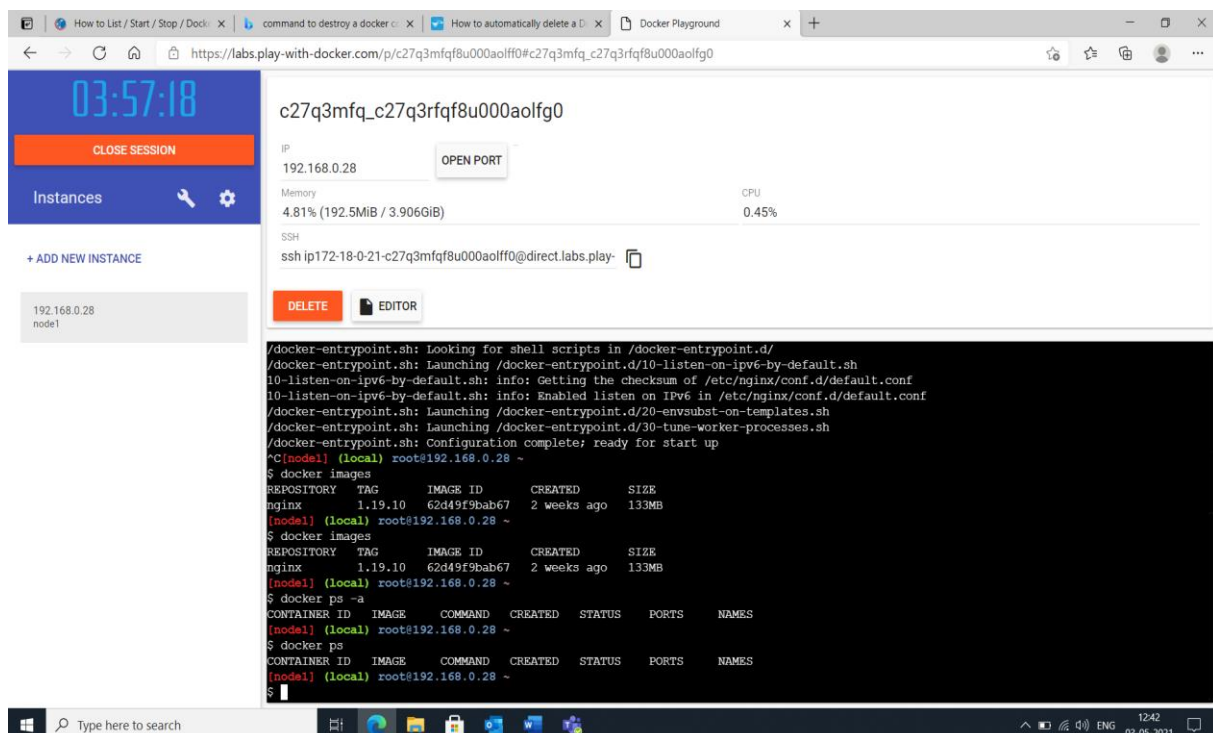
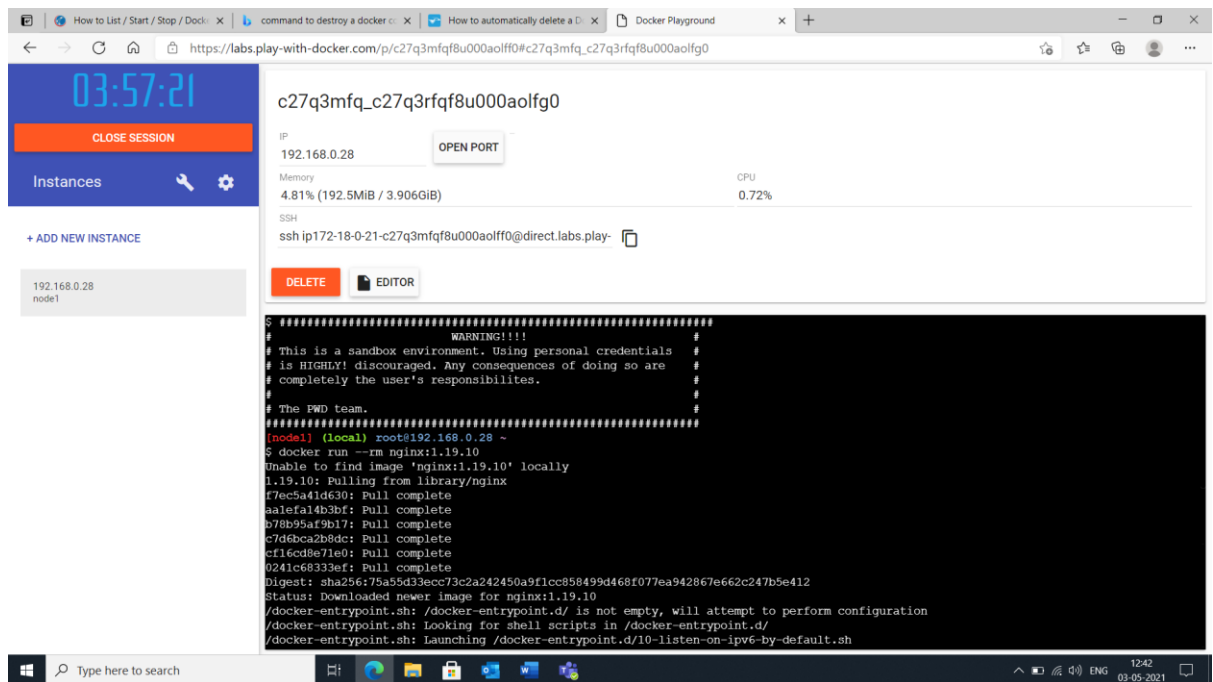


# DOCKER – EXERCISES

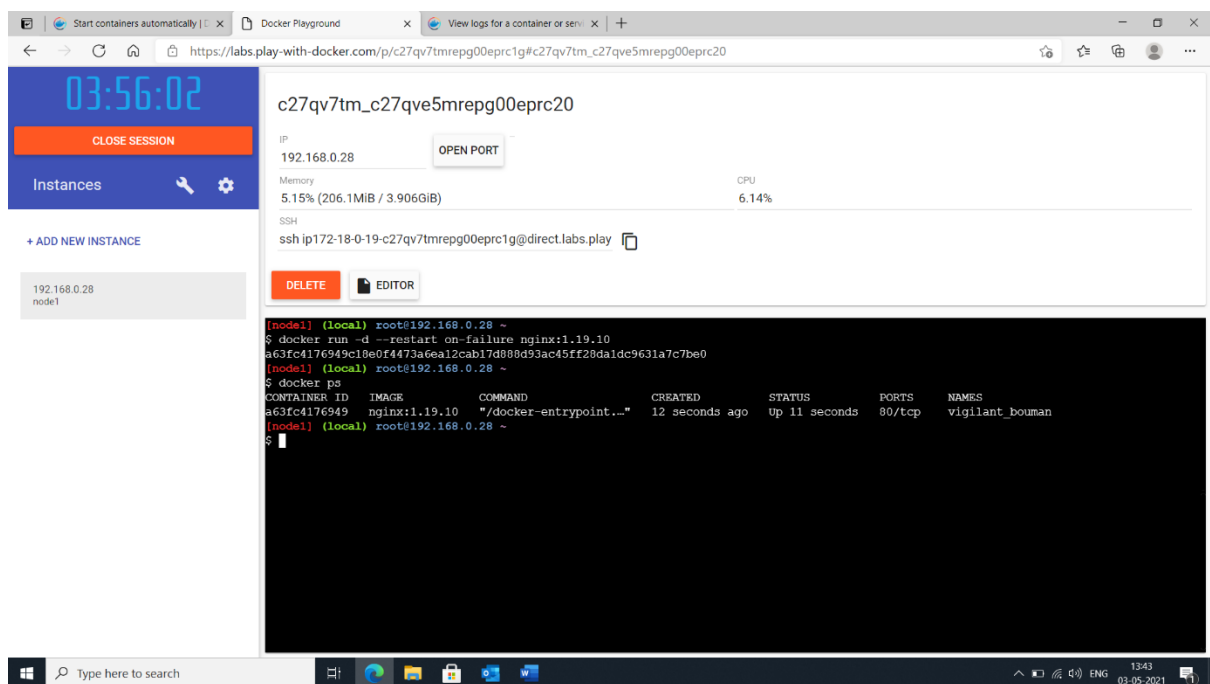
1) Spin up a temporary container with image nginx:1.19.10 and execute inside it, such that the container should be destroyed, once you exit from the container

- `docker run -rm nginx:1.19.10`
- `docker ps -a`



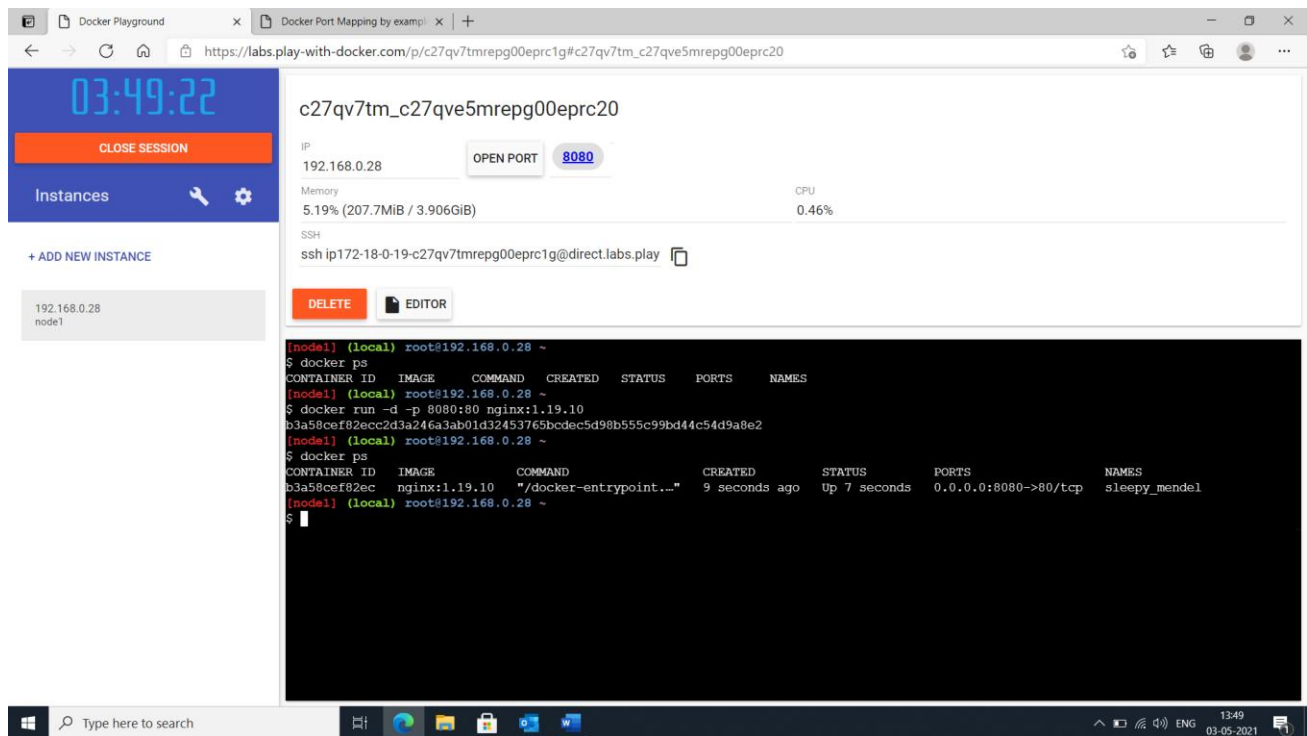
2) Spin up a container with image `nginx:1.19.10` such that it should restart automatically if any fatal errors are encountered

- `docker run -d --restart on-failure nginx:1.19.10`
- `docker ps`



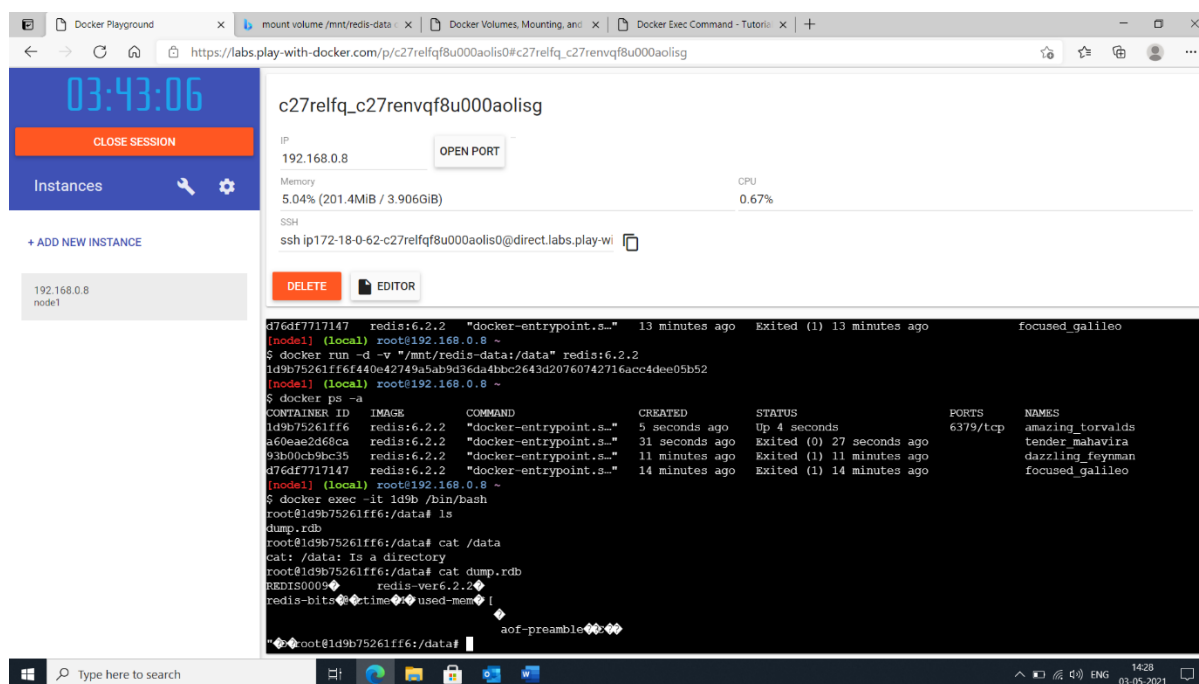
3) Spin up a container with image `nginx:1.19.10` such that port 80 of the container can be connected from port 8080 of the host

- `docker run -d -p 8080:80 nginx:1.19.10`



4) Spin up a container with image redis:6.2.2 and mount volume /mnt/redis-data of host to the /data of the container

- `docker run -d -v "/mnt/redis-data: /data " redis:6.2.2`
- `docker exec -it #container_id /bin/bash`



5) Create a dockerfile with base image centos:7, and build an image with any sample application file

▪ vi app.py

```
from flask import Flask

app = Flask(__name__)

@app.route('/')

def disp():

    return "This is Sakthi Sri's sample application"
```

▪ vi Dockerfile

# Dockerfile starts here

- ✓ FROM centos:7
- ✓ LABEL "about" = "This is a sample application file"
- ✓ RUN yum -y update
- ✓ RUN yum -y install python

- ✓ RUN yum -y install epel-release && yum clean all
- ✓ RUN yum -y install python-pip
- ✓ RUN pip install flask
- ✓ COPY ./app.py /app.py
- ✓ CMD ["echo", "Dockerfile copied"]
- ✓ ENTRYPOINT FLASK\_APP = /app.py flask run - -host 0.0.0.0

# Dockerfile ends here

- docker build -t my\_sample\_app -f Dockerfile
- docker images
- docker run -p 8080:5000 -d - --name=my\_sample\_application my\_sample\_app
- docker ps

Task Hint 50:28

How many Services exist on the system?  
in the current(default) namespace

2

3

4

0

1

Terminal 1

```

root@controlplane:~# vi Dockerfile
root@controlplane:~# vi app.py
root@controlplane:~# docker build . -t my_sample_app -f Dockerfile
Sending build context to Docker daemon 186.9kB
Step 1/10 : FROM centos:7
7: Pulling from library/centos
2d473b07cdd5: Pull complete
Digest: sha256:0f4ec88e21daf75124b8a9e5ca03c37a5e937e0e108a255d890492430789b60e
Status: Downloaded newer image for centos:7
--> 8652b9f0cb4c
Step 2/10 : LABEL "about" = "This is a sample application file"
--> Running in df54c0345e35
Removing intermediate container df54c0345e35
--> 967c14a94622
Step 3/10 : RUN yum -y update
--> Running in 2cad91b64515
Loaded plugins: fastestmirror, ovl
Determining fastest mirrors
 * base: repos.forethought.net
 * extras: mirror.es.its.nyu.edu
 * updates: centos.mirror.ltn.net

```

The screenshot shows a web browser window with the URL <https://kodekloud.com/courses/1120660/lectures/24008540>. The page displays a task titled "How many Services exist on the system? in the current(default) namespace" with a hint and a timer set to 44:37. The terminal window shows the following commands and output:

```

Downloading https://files.pythonhosted.org/packages/d2/3d/fa76db83b75c4f8d338c2fd15c8d33fdd7ad23a9b5e57eb6c5de26
b430e/click-7.1.2-py2.py3-none-any.whl (82kB)
Collecting Werkzeug>=0.15 (from flask)
Downloading https://files.pythonhosted.org/packages/cc/94/5f7079a0e0bd6863ef8f1da638721e9da21e5bacee597595b318f7
1d62e/Werkzeug-1.0.1-py2.py3-none-any.whl (298kB)
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10.1->flask)
Downloading https://files.pythonhosted.org/packages/fb/40/f3adb7cf24a8012813c5edb20329eb22d5d8e2a0ecf73d21d6b8586
5da11/MarkupSafe-1.1.1-cp27-cp27mu-manylinux1_x86_64.whl
Installing collected packages: MarkupSafe, Jinja2, itsdangerous, click, Werkzeug, flask
Successfully installed Jinja2-2.11.3 MarkupSafe-1.1.1 Werkzeug-1.0.1 click-7.1.2 flask-1.1.2 itsdangerous-1.1.0
You are using pip version 8.1.2, however version 21.1.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Removing intermediate container 135d16d4691c
--> d756069a0d50
Step 8/10 : COPY ./app.py /app.py
--> c4e8051338f7
Step 9/10 : CMD ["echo","Dockerfile copied"]
--> Running in eb79f9836453
Removing intermediate container eb79f9836453
--> a0a623a7c6ee
Step 10/10 : ENTRYPOINT FLASK_APP = /app.py flask run --host 0.0.0.0
--> Running in fb314a9fa520
Removing intermediate container fb314a9fa520
--> 0bfc497a4bf9
Successfully built 0bfc497a4bf9
Successfully tagged my_sample_app:latest
root@controlplane:~# docker run -p 8080:5000 -d --name=my_sample_application my_sample_app
unknown flag: -p
See 'docker run --help'.
root@controlplane:~#
  
```

The screenshot shows a web browser window with the URL <https://kodekloud.com/courses/1120660/lectures/24008540>. The page displays a task titled "How many Services exist on the system? in the current(default) namespace" with a hint and a timer set to 43:28. The terminal window shows the following commands and output:

```

root@controlplane:~# docker run -p 8080:5000 -d --name=my_sample_application my_sample_app
54f3f87ebd6aa99d1263d0d395a8b715186fb58f77ba42f01eebab82ef5fead4
docker: Error response from daemon: driver failed programming external connectivity on endpoint my_sample_applicati
on (0518b148114e73ab1b6280ff81c752d0b09c953b7b2522209e8f5f85727cb6aa): Error starting userland proxy: listen tcp 0.
0.0.0:8080: bind: address already in use.
root@controlplane:~# docker run -p 8080:8080 -d --name=my_sample_application my_sample_app
docker: Error response from daemon: Conflict. The container name "/my_sample_application" is already in use by cont
ainer "54f3f87ebd6aa99d1263d0d395a8b715186fb58f77ba42f01eebab82ef5fead4". You have to remove (or rename) that conta
iner to be able to reuse that name.
See 'docker run --help'.
root@controlplane:~#
  
```

6) Create a bridge network called test-app and spin up nginx and redis containers in that network

- `docker network ls`
- `docker network create -d bridge my-bridge`
- `docker images`
- `docker network inspect my-bridge`
- `docker network ls`
- `docker run -d nginx`
- `docker run -d redis`
- `docker ps`
- `docker images`
- `docker network connect my-bridge #container_id_of_nginx`
- `docker network connect my-bridge #container_id_of_redis`
- `docker network inspect my-bridge`

The screenshot shows the Docker Playground interface. On the left, there's a sidebar with a clock showing 03:58:22, a 'CLOSE SESSION' button, and an 'Instances' section with a '+ ADD NEW INSTANCE' button. Below that, a list of instances shows '192.168.0.8 node1'. The main area displays details for a session named 'c27tscfn\_c27tsevnjsv000ap7ka0'. It shows the IP '192.168.0.8', memory usage '0.94% (37.68MiB / 3.906GiB)', and CPU usage '6.42%'. There's an 'OPEN PORT' button and an SSH command: 'ssh ip172-18-0-77-c27tscfnjsv000ap7k9g@direct.labs.play-'. Below this are 'DELETE' and 'EDITOR' buttons. A terminal window is open at the bottom, showing the following commands and output:

```
[node1] (local) root@192.168.0.8 ~
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
a3759e9cac20        bridge             bridge              local
7228e48f7bcd        host              host                local
ladda33cia2e        none              null                local
[node1] (local) root@192.168.0.8 ~
$ docker network create -d bridge my-bridge
5773bd8287ce14a94c2f32e9ab58ac44da4dddbc612a0b2a5b4e8389ebfff465
[node1] (local) root@192.168.0.8 ~
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
a3759e9cac20        bridge             bridge              local
7228e48f7bcd        host              host                local
5773bd8287ce        my-bridge          bridge              local
ladda33cia2e        none              null                local
[node1] (local) root@192.168.0.8 ~
$
```

The screenshot shows the Docker Playground interface. On the left, there's a sidebar with a clock showing 03:56:51, a 'CLOSE SESSION' button, and an 'Instances' section with a '+ ADD NEW INSTANCE' button. Below that, a list of instances shows '192.168.0.8 node1'. The main area displays details for a session named 'c27u0c5m\_c27u0jtmrepg00epri60'. It shows the IP '192.168.0.8', memory usage '6.35% (253.9MiB / 3.906GiB)', and CPU usage '0.41%'. There's an 'OPEN PORT' button and an SSH command: 'ssh ip172-18-0-77-c27u0c5mrepg00epri3g@direct.labs.play-'. Below this are 'DELETE' and 'EDITOR' buttons. A terminal window is open at the bottom, showing the following commands and output:

```
$ docker images
[node1] (local) root@192.168.0.8 ~
$ docker network inspect my-bridge
{
  "Name": "my-bridge",
  "Id": "e37b0f4e5f5eb6219e32a321378475e8ba4234787ce4a01d3a71f55d6c47747a",
  "Created": "2021-05-03T11:38:17.22428742Z",
  "Scope": "local",
  "Driver": "bridge",
  "EnableIPv6": false,
  "IPAM": {
    "Driver": "default",
    "Options": {},
    "Config": [
      {
        "Subnet": "172.19.0.0/16",
        "Gateway": "172.19.0.1"
      }
    ]
  },
  "Internal": false,
  "Attachable": false,
}
```



The screenshot shows the Docker Playground interface. On the left, there's a sidebar with a clock showing 03:56:48, a 'CLOSE SESSION' button, and an 'Instances' section with a list of instances including '192.168.0.8 node1'. The main area displays details for a new instance 'c27u0c5m\_c27u0jtmrepg00eprl60'. It shows the IP address 192.168.0.8, memory usage at 6.35% (253.9MiB / 3.906GiB), and CPU usage at 0.18%. There are buttons for 'OPEN PORT', 'DELETE', and 'EDITOR'. Below this, a terminal window is open, showing a JSON configuration for a network and a shell prompt.

```

{
  "Driver": "default",
  "Options": {},
  "Config": {
    {
      "Subnet": "172.19.0.0/16",
      "Gateway": "172.19.0.1"
    }
  }
},
"Internal": false,
"Attachable": false,
"Ingress": false,
"ConfigFrom": {
  "Network": ""
},
"ConfigOnly": false,
"Containers": {},
"Options": {},
"Labels": {}
}
(node1) (local) root@192.168.0.8 ~
$
  
```

This screenshot shows the same Docker Playground instance 'c27u0c5m\_c27u0jtmrepg00eprl60' at 03:55:32. The terminal window now shows a series of Docker commands and their outputs. It lists running containers, connects them to a custom bridge network, and lists the available networks.

```

(node1) (local) root@192.168.0.8 ~
$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS    NAMES
32668ac3b6bc   redis    "/docker-entrypoint.s..." 2 minutes ago    Up 2 minutes    6379/tcp    compassionate_booth
2537157c6d0a   nginx    "/docker-entrypoint..." 3 minutes ago    Up 3 minutes    80/tcp      great_villani
(node1) (local) root@192.168.0.8 ~
$ docker network connect my-bridge 3266
(node1) (local) root@192.168.0.8 ~
$ docker network connect my-bridge 2537
(node1) (local) root@192.168.0.8 ~
$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
2d5f76c257a6   bridge   bridge    local
29ef6e53f0f4   host     host      local
e37b0f4e5f56   my-bridge bridge    local
85597d3b236d   none     null      local
(node1) (local) root@192.168.0.8 ~
$
  
```

The screenshot displays the Docker Playground web interface. On the left sidebar, there's a clock showing 03:55:06, a 'CLOSE SESSION' button, and an 'Instances' section with a '+ ADD NEW INSTANCE' button. Below this, a list shows an instance named 'node1' with IP '192.168.0.8'. The main content area is titled 'c27u0c5m\_c27u0jtmregg00eprl60' and shows the container's IP (192.168.0.8), memory usage (6.35% of 3.906GiB), and CPU usage (0.15%). It also provides an SSH command: 'ssh ip172-18-0-77-c27u0c5mregg00eprl3g@direct.labs.play'. Below this are 'DELETE' and 'EDITOR' buttons. A large terminal window at the bottom shows the container's configuration in JSON format, including container ID, name, endpoint ID, MAC address, and IP addresses. The terminal prompt is '(node1) (local) root@192.168.0.8 ~\$'.

```

"ConfigOnly": false,
"Containers": [
  "2537157c6d0a1ba7a7b1923faf2fec7f728e6c43267dfd59eeb18882428908f6": {
    "Name": "great_villani",
    "EndpointID": "295c5bc25f6be455255357e07aff79ac65481507e690c04cf4f4c0eea42b2f6b",
    "MacAddress": "02:42:ac:13:00:03",
    "IPv4Address": "172.19.0.3/16",
    "IPv6Address": ""
  },
  "32668ac3b6bc4ccfc01335ed3e964a7910f276260fd92e3193762f558f01d5f1": {
    "Name": "compassionate_booth",
    "EndpointID": "d474d67190228975ae94c672e28475a44b8f5388ab4d25fa4135b4bed88dce73",
    "MacAddress": "02:42:ac:13:00:02",
    "IPv4Address": "172.19.0.2/16",
    "IPv6Address": ""
  }
],
"Options": {},
"Labels": {}
}
(node1) (local) root@192.168.0.8 ~$
  
```