Working with Containers

Command Reference:
https://gist.github.com/initcron/0810814143889525 2de8

Objectives

- Learn how to work with containers
- Launching a Web App with existing image
- Network Port Mapping
- Container Operations e.g. inspecting, checking stats, file copying, removing etc.

In the last session we created a few containers running hello world app. We are now going to look at more practical example.

We will launch a container from an existing image and start a web application.

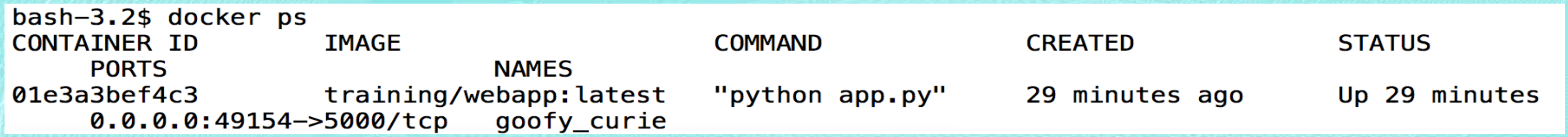
Launching Jenkins

```
$ docker run -idt -P jenkins:2.19.1-alpine
```

port mapping

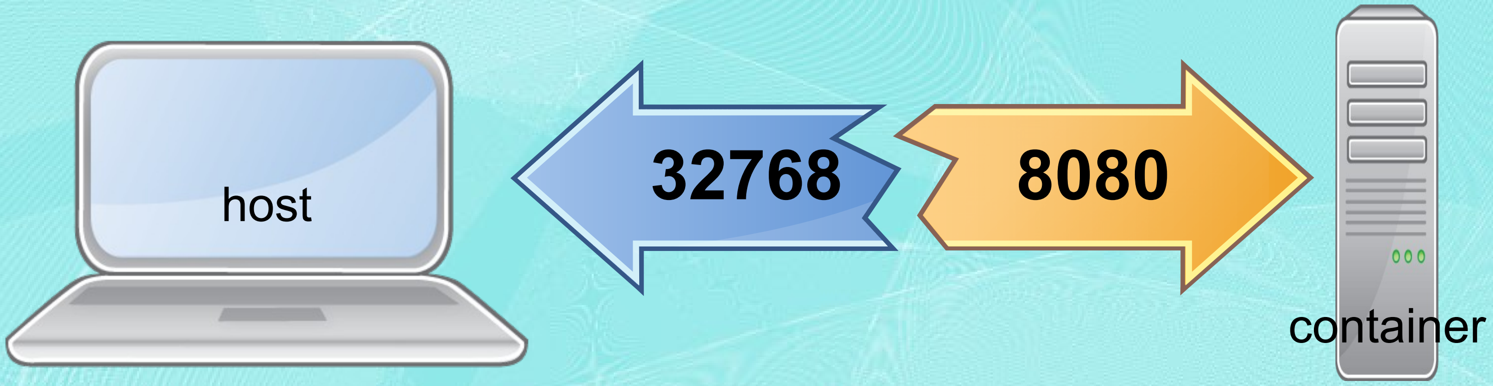
Checking Status

```
$ docker ps
```



```
bash-3.2$ docker ps
CONTAINER ID        IMAGE                            COMMAND             CREATED             STATUS
     PORTS                     NAMES
01e3a3bef4c3        training/webapp:latest   "python app.py"     29 minutes ago      Up 29 minutes
     0.0.0.0:49154->5000/tcp   goofy_curie
```

Port Mapping

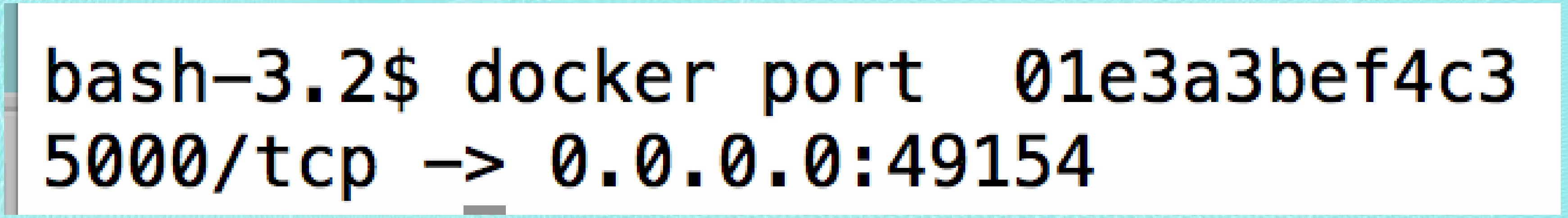


Adding -P option maps port 8080 to port 32768 on our host

-P == -p 8080

Discovering Port Mapping

```
$ docker port  <container_id>
```

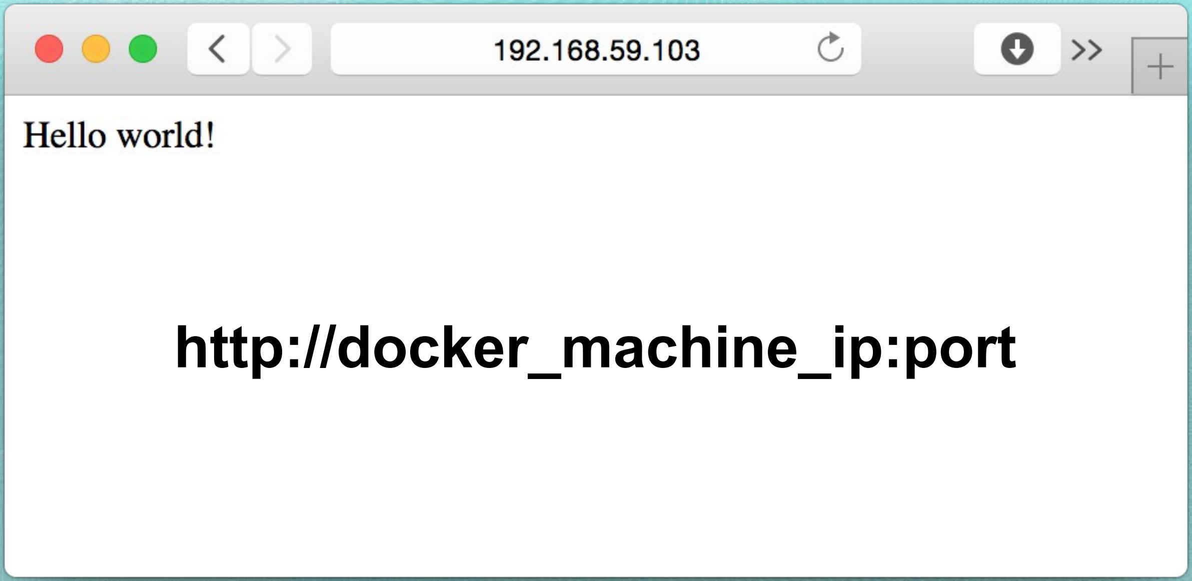


```
bash-3.2$ docker port  01e3a3bef4c3
5000/tcp -> 0.0.0.0:49154
```

This port is mapped to the VM and not to host directly. To find out the ip of the VM run the following command (on win/mac host)

```
$ docker-machine  ip default
```

Validate


192.168.59.103

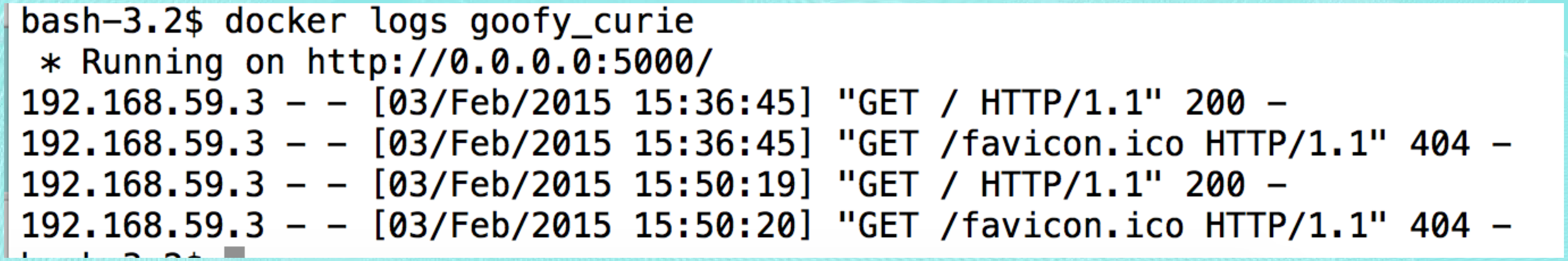
Hello world!

http://docker_machine_ip:port

http://192.168.99.100:32768

Checking Logs

```
$ docker logs <container_id>
```



```
bash-3.2$ docker logs goofy_curie
 * Running on http://0.0.0.0:5000/
192.168.59.3 - - [03/Feb/2015 15:36:45] "GET / HTTP/1.1" 200 -
192.168.59.3 - - [03/Feb/2015 15:36:45] "GET /favicon.ico HTTP/1.1" 404 -
192.168.59.3 - - [03/Feb/2015 15:50:19] "GET / HTTP/1.1" 200 -
192.168.59.3 - - [03/Feb/2015 15:50:20] "GET /favicon.ico HTTP/1.1" 404 -
```

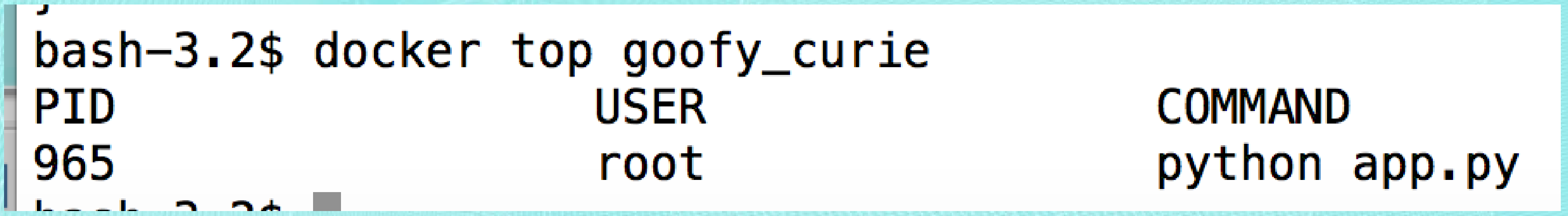
Renaming a Container

```
$ docker rename
<container_id> jenkins
```

```
$ docker ps
```

Show Process Table

```
$ docker top <container_id>
```



```
bash-3.2$ docker top goofy_curie
PID                 USER                COMMAND
965                 root                python app.py
```

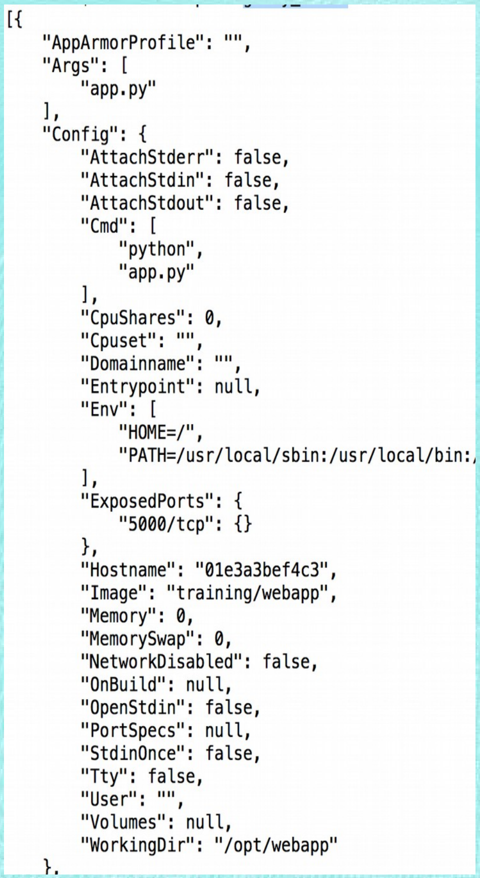
Attach to a Container

```
$ docker attach jenkins
```

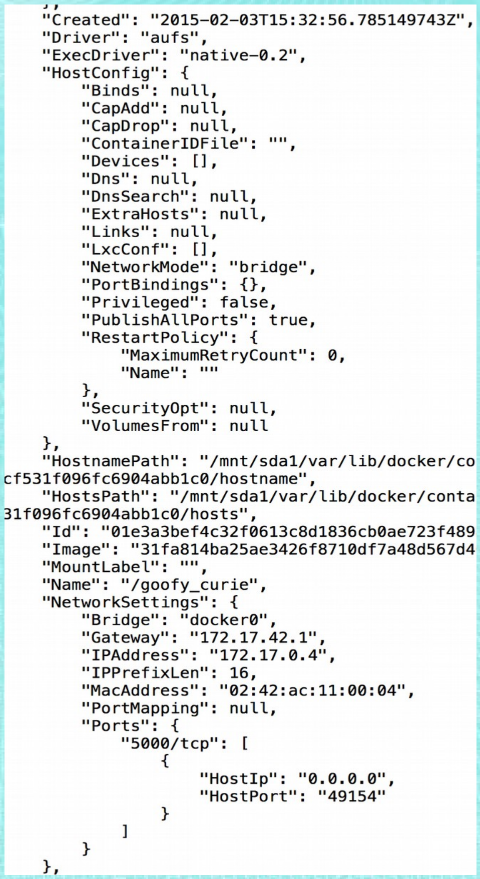
```
  To detach:

  ctrl +p , ctrl +q
```

Inspecting a Container

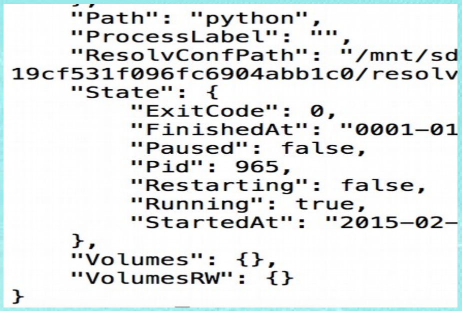
```
$ docker inspect <container_id>
```



```
[{
    "AppArmorProfile": "",
    "Args": [
        "app.py"
    ],
    "Config": {
        "AttachStderr": false,
        "AttachStdin": false,
        "AttachStdout": false,
        "Cmd": [
            "python",
            "app.py"
        ],
        "CpuShares": 0,
        "Cpuset": "",
        "Domainname": "",
        "Entrypoint": null,
        "Env": [
            "HOME=/",
            "PATH=/usr/local/sbin:/usr/local/bin:/
        ],
        "ExposedPorts": {
            "5000/tcp": {}
        },
        "Hostname": "01e3a3bef4c3",
        "Image": "training/webapp",
        "Memory": 0,
        "MemorySwap": 0,
        "NetworkDisabled": false,
        "OnBuild": null,
        "OpenStdin": false,
        "PortSpecs": null,
        "StdinOnce": false,
        "Tty": false,
        "User": "",
        "Volumes": null,
        "WorkingDir": "/opt/webapp"
    },
```



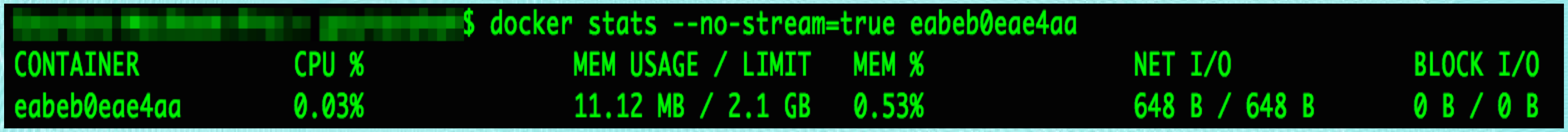
```
    "Created": "2015-02-03T15:32:56.785149743Z",
    "Driver": "aufs",
    "ExecDriver": "native-0.2",
    "HostConfig": {
        "Binds": null,
        "CapAdd": null,
        "CapDrop": null,
        "ContainerIDFile": "",
        "Devices": [],
        "Dns": null,
        "DnsSearch": null,
        "ExtraHosts": null,
        "Links": null,
        "LxcConf": [],
        "NetworkMode": "bridge",
        "PortBindings": {},
        "Privileged": false,
        "PublishAllPorts": true,
        "RestartPolicy": {
            "MaximumRetryCount": 0,
            "Name": ""
        },
        "SecurityOpt": null,
        "VolumesFrom": null
    },
    "HostnamePath": "/mnt/sda1/var/lib/docker/co
cf531f096fc6904abb1c0/hostname",
    "HostsPath": "/mnt/sda1/var/lib/docker/conta
31f096fc6904abb1c0/hosts",
    "Id": "01e3a3bef4c32f0613c8d1836cb0ae723f489
    "Image": "31fa814ba25ae3426f8710df7a48d567d4
    "MountLabel": "",
    "Name": "/goofy_curie",
    "NetworkSettings": {
        "Bridge": "docker0",
        "Gateway": "172.17.42.1",
        "IPAddress": "172.17.0.4",
        "IPPrefixLen": 16,
        "MacAddress": "02:42:ac:11:00:04",
        "PortMapping": null,
        "Ports": {
            "5000/tcp": [
                {
                    "HostIp": "0.0.0.0",
                    "HostPort": "49154"
                }
            ]
        }
    },
```



```
    "Path": "python",
    "ProcessLabel": "",
    "ResolvConfPath": "/mnt/sd
19cf531f096fc6904abb1c0/resolv
    "State": {
        "ExitCode": 0,
        "FinishedAt": "0001-01
        "Paused": false,
        "Pid": 965,
        "Restarting": false,
        "Running": true,
        "StartedAt": "2015-02-
    },
    "Volumes": {},
    "VolumesRW": {}
}
```

Show Run Stats

```
$ docker stats
$ docker stats --no-stream=true
```



```
[redacted]$ docker stats --no-stream=true eabeb0eae4aa
CONTAINER           CPU %               MEM USAGE / LIMIT    MEM %               NET I/O             BLOCK I/O
eabeb0eae4aa        0.03%               11.12 MB / 2.1 GB    0.53%               648 B / 648 B       0 B / 0 B
```

Copy Files to and From Container

```
$ touch localfile
$ docker cp localfile jenkins:/opt/
$ docker cp
jenkins:/var/jenkins_home/config.xml .
$ docker diff jenkins
```

Docker Diff Status

- A – Add
- C – Change
- D – Delete

Controlling Resources

cgroups in action

Updating Memory Limit

```
$ docker inspect jenkins | grep
            -i memory
$ docker stats --no-stream=true

$ docker update -m 400M jenkins
$ docker stats --no-stream=true
```

Limiting Resource at Launch

Open a terminal and watch docker stats while running containers in a separate window

```
$ docker stats
```

Launch Batch 1

```
$ docker run -d --name st-01
schoolofdevops/stresstest stress --cpu 1

$ docker run -d --name st-02 -c 512
schoolofdevops/stresstest stress --cpu 1
```

Launch Batch 2

```
docker run -d --name st-03 -c 512
schoolofdevops/stresstest stress --cpu 1

docker run -d --name st-04
schoolofdevops/stresstest stress --cpu 1
```

Running Privileged Containers

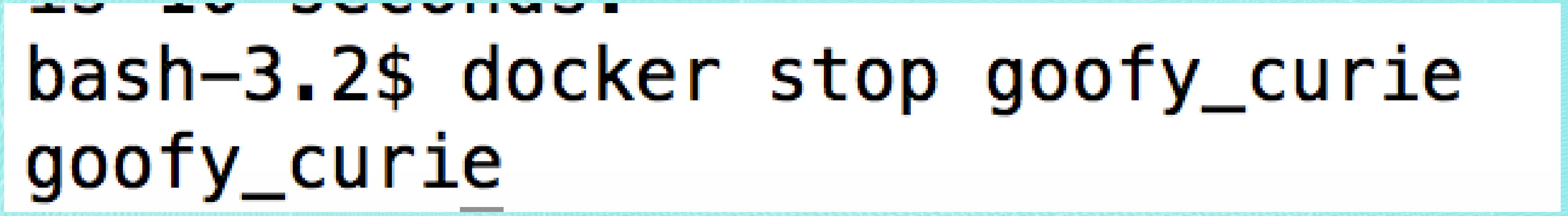
```
docker run -itd --name=sysdig --privileged=true \

--volume=/var/run/docker.sock:/host/var/run/docke
r.sock \
           --volume=/dev:/host/dev \
           --volume=/proc:/host/proc:ro \
           --volume=/boot:/host/boot:ro \

--volume=/lib/modules:/host/lib/modules:ro \
           --volume=/usr:/host/usr:ro \
           sysdig/sysdig:0.11.0 sysdig
```

Monitoring with Sysdig

```
$ docker exec -it sysdig bash
$ csysdig
```

Stopping Container

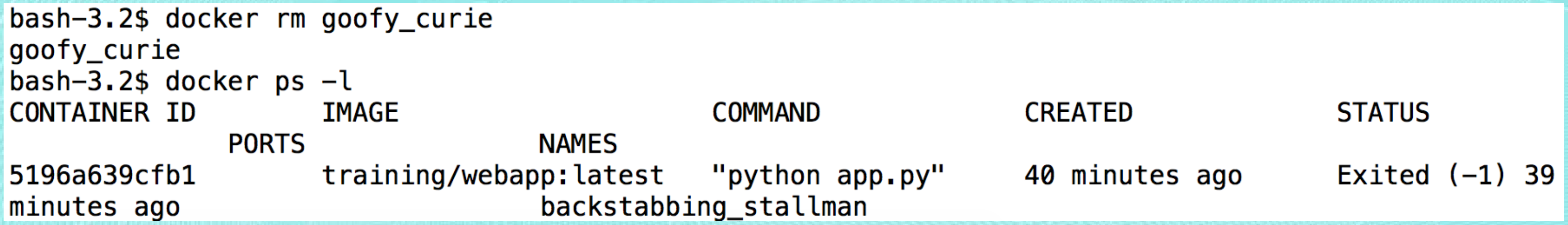
```
$ docker stop jenkins
$ docker ps -l
```



```
bash-3.2$ docker stop goofy_curie
goofy_curie
```

Removing Container

```
$ docker rm jenkins
$ docker ps -l
```



```
bash-3.2$ docker rm goofy_curie
goofy_curie
bash-3.2$ docker ps -l
CONTAINER ID        IMAGE                         COMMAND             CREATED             STATUS
              PORTS                   NAMES
5196a639cfb1        training/webapp:latest   "python app.py"     40 minutes ago      Exited (-1) 39
minutes ago                           backstabbing_stallman
```

Exercise

- Create/Run a new container and limit its memory to 300M while launching it.

- Hint: look at docker run --help for options to control resource utilization e.g. memory, cpu

Summary

- Launching Containers from pre built image
- Connecting to Applications, Port Mapping
- Container Operations