AMP Project

Tony Pujals

Director of Cloud Engineering

tpujals@appcelerator.com

appcelerator®

Please make sure to install the latest version of the Docker tools:

Mac/PC

https://www.docker.com/products/overview#/docker_toolbox

Linux

https://docs.docker.com/linux/step_one/

Agenda

- Use docker-machine to create a docker host for containers
- Dockerize a Node.js server app
    - Expose container ports
    - Map working directory to container while developing
    - Create image when finished editing
    - Debug app running in container with Node Inspector

Docker Machine

appcelerator®

docker-machine

- Tool for provisioning a host with the Docker engine to run containers
- Driver for many environments and cloud providers
- Basic format looks like this

```
$ docker-machine create -d <driver> <machine-name>
```

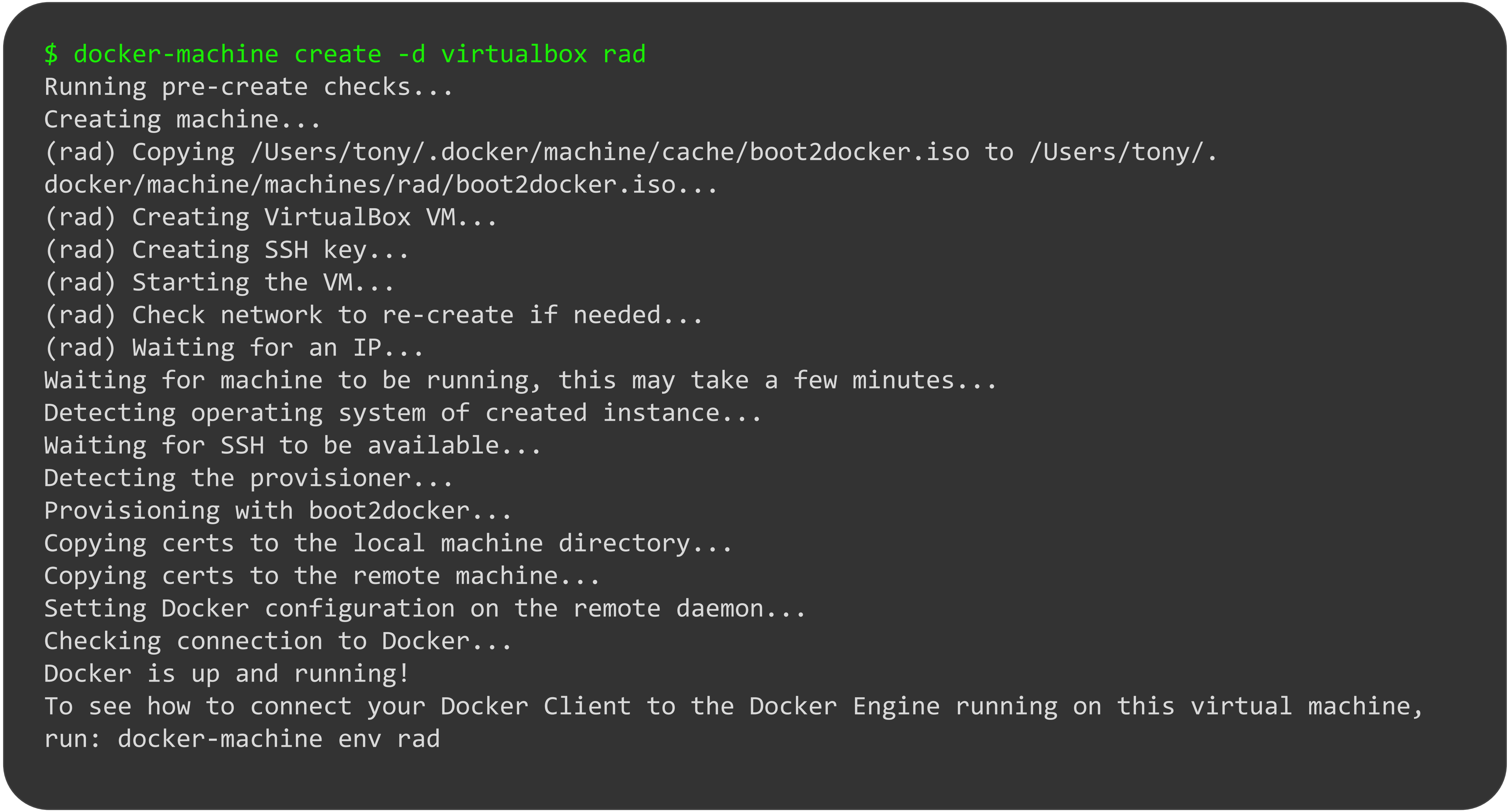
docker-machine

Host for Docker Engine must be running Linux kernel 3.10+

On a Mac/PC, that means need to run Docker hosts in VMs

- When you install Docker Toolbox it will set up VirtualBox for you
- Using the VirtualBox driver, docker-machine will provision Docker hosts as virtual machines running a lightweight Linux distro called Boot2docker

Create a machine called 'rad'...



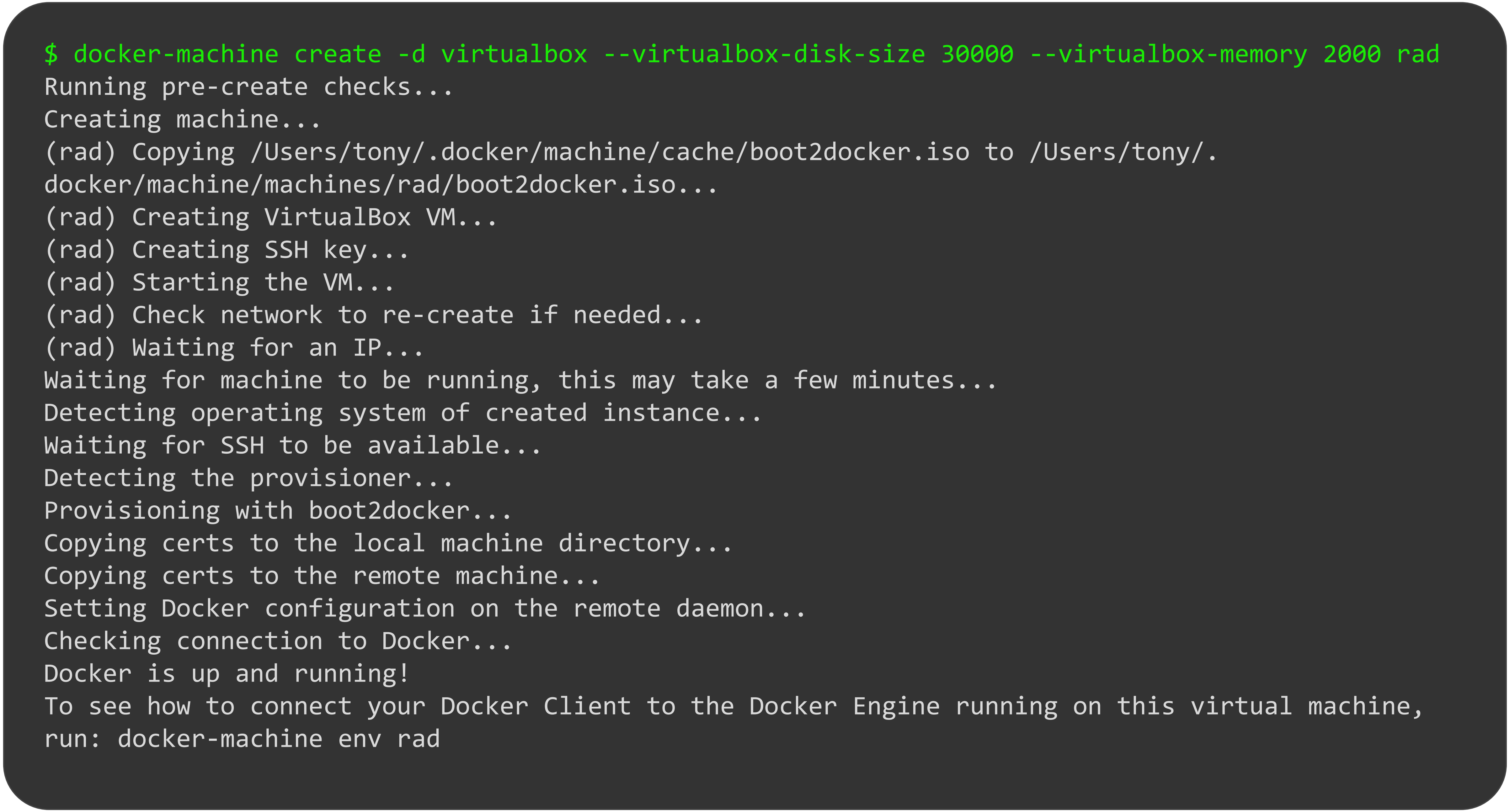
```
$ docker-machine create -d virtualbox rad
Running pre-create checks...
Creating machine...
(rad) Copying /Users/tony/.docker/machine/cache/boot2docker.iso to /Users/tony/.
docker/machine/machines/rad/boot2docker.iso...
(rad) Creating VirtualBox VM...
(rad) Creating SSH key...
(rad) Starting the VM...
(rad) Check network to re-create if needed...
(rad) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine,
run: docker-machine env rad
```

docker-machine

You can pass provider-specific options to the driver

Enter the command without the machine name argument to see options

Common options for virtualbox driver



```
$ docker-machine create -d virtualbox --virtualbox-disk-size 30000 --virtualbox-memory 2000 rad
Running pre-create checks...
Creating machine...
(rad) Copying /Users/tony/.docker/machine/cache/boot2docker.iso to /Users/tony/.
docker/machine/machines/rad/boot2docker.iso...
(rad) Creating VirtualBox VM...
(rad) Creating SSH key...
(rad) Starting the VM...
(rad) Check network to re-create if needed...
(rad) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine,
run: docker-machine env rad
```

docker-machine

Common commands

- `docker-machine ls`
- `docker-machine active`
- `docker-machine start [name]`
- `docker-machine restart [name]`
- `docker-machine stop [name]`
- `docker-machine env [name]`
- `docker-machine ip [name]`
- `docker-machine inspect [name]`
- `docker-machine rm [name]`

docker-machine

Get command line help:

- docker-machine help [command]
- docker-machine [command] --help

docker-machine

Remember:

`docker-machine` commands apply to hosts

`docker` commands apply to things you can do on a specific hosts

docker-machine

When working with multiple machines, set the environment so **docker** communicates with the correct machine

docker-machine env [name] exports environment variables that **docker** uses

```
$ eval $(docker-machine env <machine-name>)
```

docker-machine

`docker-machine env [name]` by itself writes environment variables to **`stdout.`**

```
$ docker-machine env rad
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://192.168.99.102:2376"
export DOCKER_CERT_PATH="/Users/tony/.docker/machine/machines/rad"
export DOCKER_MACHINE_NAME="rad"
# Run this command to configure your shell:
# eval $(docker-machine env rad)
```

The **`eval`** command is used to evaluate commands in the current shell.

```
$ eval $(docker-machine env rad)
```

docker-machine

When on a Mac/PC, containers run in a virtual machine, not localhost.

Use **`docker-machine ip [name]`** to get the IP address of the host.

```
$ docker-machine ip rad
192.168.99.102
```

Dockerizing a Node app

appcelerator®

`express-01` is a simple node app

`express-02` has a Dockerfile and docker-compose file.

express-02: Dockerfile

```
FROM atomiq/node:onbuild

# helpful while debugging, will remove in the future
RUN apt-get update && apt-get install -y vim

# server app
EXPOSE 3000

CMD ["node", "app.js"]
```

express-02: docker-compose.yml

```
version: '2'
services:
  web:
    build: .
    image: express-02
    ports:
      - "3000"
    environment:
      DEBUG: app*,-*
      NODE_ENV: development
    command:
      [ "node", "app.js" ]
```

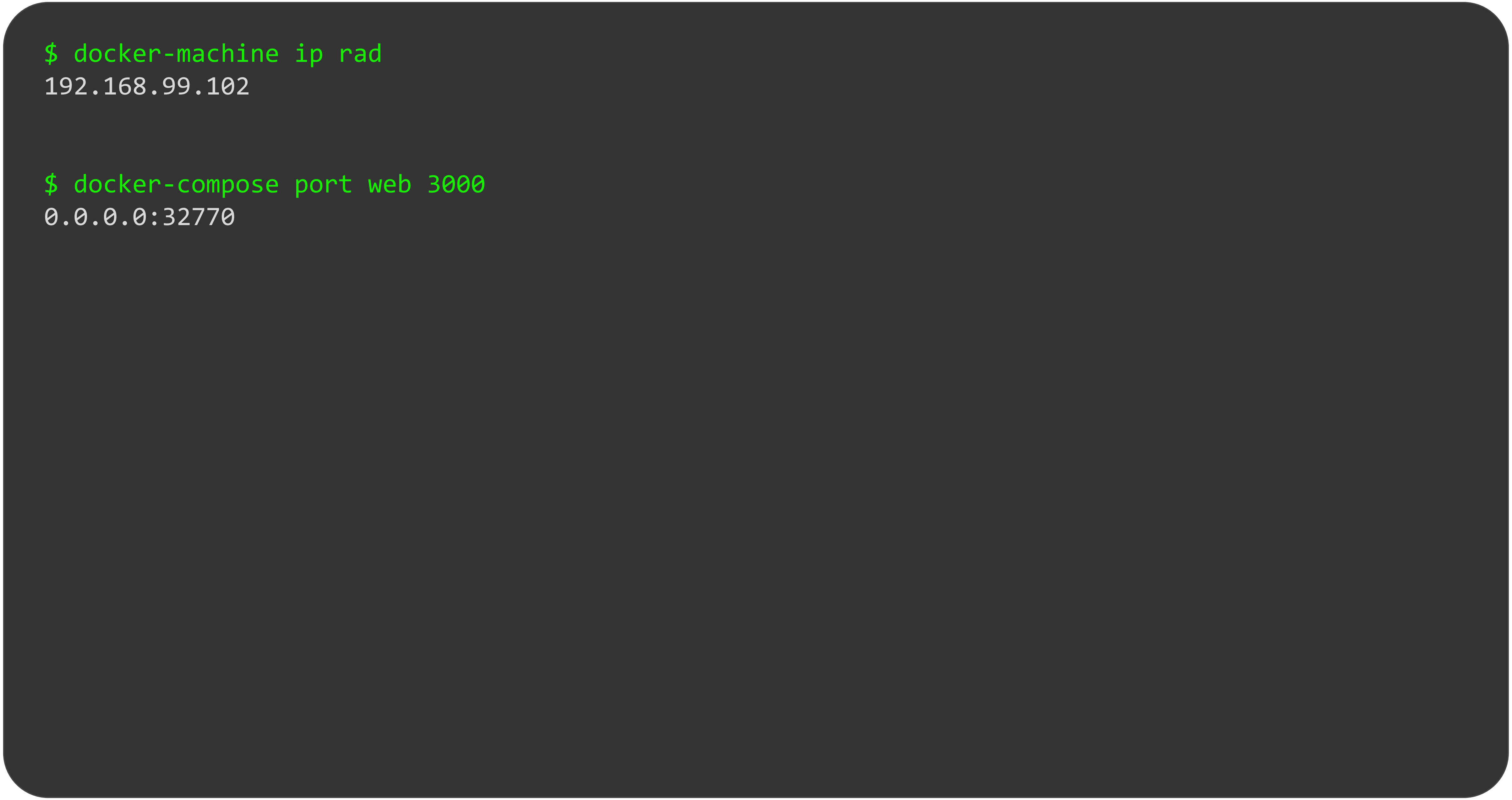
Run the app in a container
with **docker-compose up**

```
$ docker-compose up
Creating network "express02_default" with the default driver
Building web
Step 1 : FROM atomiq/node:onbuild
onbuild: Pulling from atomiq/node
fdd5d7827f33: Pull complete
a3ed95caeb02: Pull complete
0f35d0fe50cc: Pull complete
627b6479c8f7: Pull complete
67c44324f4e3: Pull complete
b52090f6ca00: Pull complete
187b1e17550c: Pull complete
190708e4e099: Pull complete
Digest: sha256:a3317bc1b070bfef0744d6b20910e0ed3ce11eae14173aa26b881f2e03a51a62
Status: Downloaded newer image for atomiq/node:onbuild
# Executing 3 build triggers...
…
```

Need two pieces of information to connect to the app

```
docker-machine ip rad
```

```
docker-compose port web 3000
```



```
$ docker-machine ip rad
192.168.99.102


$ docker-compose port web 3000
0.0.0.0:32770
```

```
function addr {
  service=$1
  containerport=$2
  file=${3:-"docker-compose.yml"}
  echo $(docker-machine ip $(docker-machine active)):$(docker-compose -f $file port $service
$containerport | cut -d ':' -f 2
}
```

https://gist.github.com/subfuzion/41b8e2b9eed34532f910 `express-03` mounts the current directory so the image doesn't need to be rebuilt during development

3 compose files are used

- **docker-compose.base.yml**
- **docker-compose.prod.yml**
- **docker-compose.yml**

docker-compose.base.yml

```yaml
version: '2'
services:
  web:
    build: .
    image: express-03
    ports:
      - "3000"
    environment:
      DEBUG: app*,-express*
      NODE_ENV: development
    command:
      [ "node", "app.js" ]
```

docker-compose.yml

```yaml
version: '2'
services:
  web:
    extends:
      file: docker-compose.base.yml
      service: web
    environment:
      DEBUG: app*,-express*
      NODE_ENV: development
    volumes:
      - ./:/usr/src/app
```

docker-compose.prod.yml

```
version: '2'
services:
  web:
    extends:
      file: docker-compose.base.yml
      service: web
    environment:
      NODE_ENV: production
```

`express-04` adds a docker-compose.debug.yml to support Node Inspector docker-compose.debug.yml

```
version: '2'
services:
  web:
    extends:
      file: docker-compose.yml
      service: web
    ports:
      - "8080:8080"
    command: "node_modules/.bin/node-debug --no-preload --web-host 0.0.0.0 ./app.js"
```

```
$ IP=$(docker-machine ip rad)
$ open http://$IP:8080/?ws=$IP:8080&port=5858
```

Where to go next

https://github.com/appcelerator/amp-project/wiki/Docker

✉ tpujals@appcelerator.com

@subfuzion
https://twitter.com/subfuzion

in https://www.linkedin.com/in/tonypujals/