THE LIST RESOURCES DEPLOYED FOR THIS PROJECT:
Docker
Vagrant
Nginx
Ubuntu 14.04 (Trusty Tahr)
Supervisor

CORRIGENDUM:
1. Several attempts to my connect to my Hosted-Chef account @ OPSCODE failed. Attempts to invoke the **knife.rb** resulted in several **"too many symbolic links"** errors. It appears the many UBUNTU 14.04 (LTS) users in groups such as "*Stackoverflow*" have reported and documented similar issues with Chef 12.
e.g. https://github.com/chef/issues/2677.

2. Oracle VirtualBox (with GuestAdditions). I could not find stable (bundled) images to work with Vagrant on Ubuntu 14.04 LTS; at every attempt at connecting to the downloaded images kept timing out during initial configuration. Several attempts, with dozens of different O/S images were to no avail.

3. Even if a "stable" OracleVirtualBox (with GuestAdditions) image had been used, it would have needlessly added an extra layer of complexity in putting the pieces together, resulting in something that will look like: LocalHost --> OracleVirtualBox --> Vagrant --> Docker.

4. As stable as Ubuntu 14.04 LTS (Trusty Tahr) is, it appears to have known problems compiling the ruby/gem libraries needed for knife to work without problems. At any rate, it appears however that **Chef** recommends Ubuntu 12.04 LTS (Precise Pangolin) in order to with Chef Server 12. Unfortunately, changing the version of Ubuntu on my workstation was not possible.

5. A docker container is alive only if an active process is running within it. However, as long as the processor manager **Supervisor** daemon continues to run, the container will continue to as well.

PROJECT

Install Vagrant and create a configuration file - **Vagrantfile** - for this project in the local directory where this project will be executed.

```
config.vm.box_url =
"http://storage.core-os.net/coreos/amd64-generic/dev-channel/coreos_production_vagrant.bo
x"
config.vm.network
"private_network",

    ip: "172.12.8.150"
```

Download and install Supervisor with apt-get install supervisor

Run docker as root, to download an Ubuntu base image from a repository.

```
kayode@Lenovo-G580:~$ sudo -s
root@Lenovo-G580:~# docker run ubuntu /bin/bash
Unable to find image 'ubuntu' locally
Pulling repository ubuntu
2d24f826cb16: Download complete
511136ea3c5a: Download complete
fa4fd76b09ce: Download complete
1c8294cc5160: Download complete
117ee323aaa9: Download complete
```

To see currently running docker processes

```
root@Lenovo-G580:~# docker ps -a
CONTAINER      ID IMAGE       COMMAND        CREATED          STATUS            PORTS    NAMES
63ac69e1f0c0   ubuntu:latest   /bin/bash      3 minutes ago    Exited (0) 3 minutes ago   determined_jones
```

A docker container is alive only if an active process is running within it. However, as long as the processor manager **Supervisor** daemon continues to run, the container will continue to as well.

Running as user_id docker, run ps -a again, shows there are two containers running, and their docker IDs

```
root@Lenovo-G580:~# docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
bc04a3068d2c ubuntu:latest /bin/bash 56 seconds ago Exited (127) 11 seconds ago lonely_ptolemy
63ac69e1f0c0 ubuntu:latest /bin/bash 18 minutes ago Exited (0) 18 minutes ago determined_jones
```

Running diff <container id>

```
kayode@Lenovo-G580:~$ docker diff bc04a3068d2c
2015/02/28 03:31:24 Get http:///var/run/docker.sock/v1.12/containers/bc04a3068d2c/changes: dial unix /var/run/docker.sock: permission denied
root@Lenovo-G580:~# docker diff bc04a3068d2c
A /.bash_history
```

Logging back into the docker container to install some utilities - this command generates a very long inventory of things it is installing.

```
docker run -t -i ubuntu /bin/bash
```

There is a need to update the O/S thereafter

```
apt-get update
```

```
apt-get install -y git ack-grep vim curl wget tmux build-essential python-software-properties
```

Do a diff to list the changed files and directories in a container's filesystem

```
root@Lenovo-G580:~# docker diff bc04a3068d2c
A /.bash_history
root@Lenovo-G580:~#
```

To commit the docker container by NAME_ID and TAG.
docker commit <Container ID> <Name>:<Tag>

```
 root@Lenovo-G580:~# docker commit bc04a3068d2c lenovo/docker-myproject:0.1
b754431e1698ae4d045273c2c4cbe2edcb5fc76682cdd1759da7b5d62e86ec36
```

To confirm that the **commit** was successful.

```
root@Lenovo-G580:~# docker images
```

```
REPOSITORY              TAG         IMAGE ID            CREATED          VIRTUAL SIZE
lenovo/docker-myproject:0.1         b754431e1698       2 minutes ago        188.3 MB
ubuntu latest 2d24f826cb16 7 days ago 188.3 MB
```

Now there is a docker image named **lenovo/docker-myproject:0.1**

To build a Server with Dockerfile the nect step is to create a directory which will be shared. In this case, I named it **compact** and I *"cd-ed"* into it, and inside it, I created a file called **default**

```
root@Lenovo-G580:~# mkdir compact
```

The file named **default**, to which I added the following lines:

```
server {
    root /var/www;
    index index.html index.htm;

    # Make site accessible from http://localhost/
    server_name localhost;

    location / {
         # First attempt to serve request as file, then
         # as directory, then fall back to index.html
           try_files $uri $uri/ /index.html;
     }
}
```

For this project, the the http sever of choice is **nginx**. It was chosen for its simplicity.
Nginx is specified in the next file - **Dockerfile** - that was created. **Dokerfile** that will be used to bt docker to configure the web server.

```
FROM lenovo/docker-myproject:0.1

RUN echo "deb http://archive.ubuntu.com/ubuntu precise main universe" > /etc/apt/sources.list
RUN apt-get update
RUN apt-get -y install nginx

RUN echo "daemon off;" >> /etc/nginx/nginx.conf
RUN mkdir /etc/nginx/ssl
ADD default /etc/nginx/sites-available/default

EXPOSE 80

CMD ["nginx"]
```

The **Dockerfile** script does the following:

-
- FROM will tell Docker what image (and tag in this case) to base this off.
-
- RUN will run the given command (as user "root") using sh -c "your-given-command"
-
- ADD will copy a file from the host machine into the container

This is handy for configuration files or scripts to run, such as a process watcher like supervisord, systemd, upstart, forever (etc)

-
- EXPOSE will expose a port to the host machine. You can expose multiple ports like so: EXPOSE 80 443 8888
-
- CMD will run a command (not using sh -c). This is usually your long-running process. In this case, we're simply starting Nginx. I want something watching the nginx process in case it fails.

Build the a new docker image from the example that was just downloaded.

```
docker build -t nginx-example .
```

```
root@Lenovo-G580:~/compact# docker images
REPOSITORY              TAG            IMAGE ID        CREATED          VIRTUAL SIZE
lenovo/docker-myproject 0.1            b754431e1698    22 minutes ago   188.3 MB
ubuntu latest 2d24f826cb16 7 days ago 188.3 MB
```

To view Docker processes running

```
root@Lenovo-G580:~/compact# docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
e1abf136a9e6 ubuntu:latest /bin/bash 15 minutes ago Exited (127) 13 minutes ago romantic_ptolemy
954b3237cf3c ubuntu:latest /bin/bash 26 minutes ago Exited (0) 26 minutes ago cranky_mccarthy
2db3c686034a ubuntu:latest /bin/bash 28 minutes ago Exited (127) 27 minutes ago prickly_meitner
8aa31d0f5b7e ubuntu:latest /bin/bash 29 minutes ago Exited (0) 29 minutes ago elegant_archimedes
24ddf8efdd40 ubuntu:latest /bin/bash 41 minutes ago Exited (127) 31 minutes ago drunk_blackwell
bc04a3068d2c ubuntu:latest /bin/bash 53 minutes ago Exited (127) 52 minutes ago lonely_ptolemy
63ac69e1f0c0 ubuntu:latest /bin/bash About an hour ago Exited (0) About an hour ago determined_jones
```

For each line in the **Dockerfile**, a new container (and commit) is produced

```
docker run -p 80:80 -d nginx-example
docker run -p 80:80 -d nginx-example
```

The *-p 80:80* binds the Container's port 80 to the guest machines, so if we curl localhost or go to the server's IP address in our browser, we'll see the results of Nginx processing requests on port 80 in the container.

```
root@Lenovo-G580:~/compact# docker run -p 80:80 -d lenovo/docker-myproject:0.1
e41d4e3cd63573b3b0e3193d76f54f48b94218642f8121705335865682f72ecc
```

To view if the Docker images is now running

```
root@Lenovo-G580:~/compact# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

root@Lenovo-G580:~/compact# docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
e41d4e3cd635 lenovo/docker-myproject:0.1 /bin/bash 2 minutes ago Exited (0) 2 minutes ago desperate_albattani
ee703b8581e9 lenovo/docker-myproject:0.1 /bin/bash 3 minutes ago Exited (0) 3 minutes ago prickly_thompson
e1abf136a9e6 ubuntu:latest /bin/bash 28 minutes ago Exited (127) 26 minutes ago romantic_ptolemy
954b3237cf3c ubuntu:latest /bin/bash 39 minutes ago Exited (0) 39 minutes ago cranky_mccarthy
```

The **localhost** (my workstation) can now be “curled” to verify if web server is running.

```
root@Lenovo-G580:~/compact# curl localhost/index.htmld
curl: (7) Failed to connect to localhost port 80: Connection refused
root@Lenovo-G580:~/compact# <html>
bash: syntax error near unexpected token `newline'
root@Lenovo-G580:~/compact# <head><title>500 Internal Server Error</title></head>
bash: syntax error near unexpected token `<'
root@Lenovo-G580:~/compact# <body bgcolor="white">
bash: syntax error near unexpected token `newline'
root@Lenovo-G580:~/compact# <center><h1>500 Internal Server Error</h1></center>
bash: syntax error near unexpected token `<'
root@Lenovo-G580:~/compact# <hr><center>nginx/1.1.19</center>
bash: syntax error near unexpected token `<'
root@Lenovo-G580:~/compact# </body>
bash: syntax error near unexpected token `newline'
root@Lenovo-G580:~/compact# </html>
```

WE GET A 500 ERROR!

That's likely because there's no default index.html file for Nginx to fall back onto. However in order to begin to correct this problem, first stop the container.

The Container was stopped with <container id>:

```
root@Lenovo-G580:~/compact# docker stop e41d4e3cd635
```

To fix this, a **share directory** between the Container and the host machine was created. First, an **index.html** page in the **share directory** share it.

```
root@Lenovo-G580:~/compact# ls -lsa
total 16
4 drwxr-xr-x 2 root root 4096 Feb 28 04:14 .
4 drwxr-xr-x 19 kayode kayode 4096 Feb 28 04:04 ..
4 -rw-r--r-- 1 root root 314 Feb 28 04:09 default
4 -rw-r--r-- 1 root root 326 Feb 28 04:14 Dockerfile
root@Lenovo-G580:~/compact# mkdir share
root@Lenovo-G580:~/compact# ls -l
total 12
-rw-r--r-- 1 root root 314 Feb 28 04:09 default
-rw-r--r-- 1 root root 326 Feb 28 04:14 Dockerfile
drwxr-xr-x 2 root root 4096 Feb 28 04:46 share
```

The Docker container is now restarted

```
docker run -v /home/kayode/compact/share:/var/www:rw -p 80:80 -d nginx-example
```

the web server will now be deployed with the following command:

```
root@Lenovo-G580:~/compact# docker run -v /home/kayode/compact/share:/var/www:rw -p 80:80 -d nginx-example
0f13c70b15b0f7fd1d48e571066dd6ab4126f8c1608acfdc935b7e6e3133dc0
root@Lenovo-G580:~/compact# curl localhost
```

In a web browser, connect to 127.0.0.1 on prt 80

Automation for the people