ECS781P: Cloud Computing
Lab Instructions for Week 3

More on OS, Virtualization, Intro to Application Containerization

Dr. Arman Khouzani

Jan 24, 2017

1 More on OS

1. **File-System** is a term describing how the OS names files and organizes them (places them logically) for storage and retrieval (to access them whenever needed). Open an terminal and change your current directory to root by typing `cd /`. Now write down the name of at least five *directories*, what they stand for, and what does Unix/Linux use them for. (Note: `usr` stands for Unix System Resources, not user!) You can get help from the *Linux Documentation Project* website: `http://www.tldp.org/index.html`. Specifically, what do we mean by the `binaries (bin)` and `libraries (lib)`?

2. We can specifically find out where each command in Linux is (logically) stored by issuing the `which` and then name of the command. Find out where the following commands are located: `ls, cp, rm, ssh, top, scp`. Did you get a feeling about where a command is likely to be found. For instance, before checking, make a guess where `rsync` and `gcc` are likely to be found. Were you correct in your guess?

3. See the content of `/usr/local/` what is being located here?

4. Check if the following commands are available on your machine, and if so, where are they stored (by issuing `which`):
   `firefox, latex, python, matlab, mathematica, qemu-img`

2 More on Virtualization

1. *Virtualization* in general means, having a software that runs on a machine and *mimics* (*emulates*) another system with the same interface, such that any program/user that interacts with it would see the same outward behaviour irrespective of what the underlying machine is.

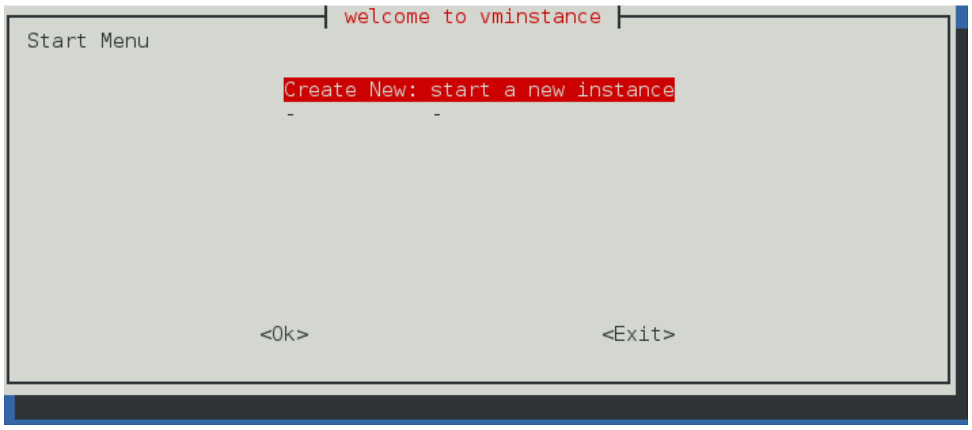
There are many types of virtualisation, depending on what is being "emulated". Find a one line explanation for each of these and an example implementation:

Server Virtualization:
Desktop Virtualization:
Network Virtualization:
Data Virtualization:
Storage Virtualization:
Application Virtualization:

2. Run `vminstance`. Delete any images that there are and you might have had from before. Now go to *Create New: start a new instance*. Choose your favourite OS from what is available. We will stick with CentOS. (Note, you will return and working with this image, so be a bit careful in your choice!).

   Caution: You should not exit the VM manager before the VM is shutdown (e.g. by running `sudo shutdown -h now` or `reboot -p` or `halt` or `poweroff` from a terminal in the VM, or the usual graphical way). Otherwise, you will lose all the changes you have made to the VM (including all the new software that you install and all your files that you develop).

3. All the new software that we install will be stored on the image if we shutdown properly. For instance, install `htop` on your instance (the instruction for installing new software were given on the last week lab sheet). Run (connect) to



   the image. Go to your home directory (`cd ~`). Make a `bin` directory and change into it.

4. We will be testing a sample `shell script`. Create a text file with your chosen name (e.g. `my_first_script`). You can create and edit text files using one of the following commands/software: `vim, nano, emacs`, but if you prefer something very user-friendly, try `gedit` or `kwrite` (or any of your favourite!). Again, remember, you have root privilege on the VM.

5. Inside the text file just write the following and save and exit:

```
#!/bin/bash

echo "Hello World!"
```

6. Give read and execute permission to all on your file (recall `chmod` was the command to use, with proper parameters).

7. execute the shell script simply by typing: `./my_first_script`.

8. Now, for your second shell script, create another text file with the following content and save it.

   ```
   #!/bin/sh

   free -m | awk 'NR==2{printf "Memory Usage: %s/%sMB (%.2f%%)\n", $3,$2,$3*100/$2 }'
   df -h | awk '$NF=="/"{printf "Disk Usage: %d/%dGB (%s)\n", $3,$2,$5}'
   top -bn1 | grep load | awk '{printf "CPU Load: %.2f\n", $(NF-2)}'
   ```

   Can you guess what it will do? Try it.

9. Edit your second shell script to show the amount of free memory in units of GB instead of MB. Try it and make sure the output is shown properly.

10. Read the manual of `scp` and `rsync`. Use one of them to copy your script files from the vm to the host machine (in your home directory). What "protocol" is being used for this transfer?

    Note: this is a good practice to do this "backup" with all the files that you will develop on your virtual machine from now on.

3 Application Containerization

Application containerization, or simply, containerization is closely related to virtual machines, but has very important distinctions.
In VM, the entire OS is emulated (including all the functionalities of the kernel). This is good when our purpose is exactly to have a working machine (desktop, or server) that is not tied up to its underlying hardware.
But if our objective is to run applications that will run on any hardware and OS, then spinning an entire virtual machine looks like a rather waste of the host resources. This is where containerization is introduced. The main implementations of the containerization idea are: `Docker, LXC, runC`. We will be using `Docker`. In brief: a virtual machine provides an abstract infrastructure while the container provides an abstract OS.

❒ Our only task for this part (today) is to make sure we have downloaded and installed `Docker` engine on our VM instances.

  Again, note: make sure you properly shutdown your vm instance after these steps, or all your installation will not be saved next time you log in.

  Note 2: The only reason we are installing Docker on the VMs is that we do not have admin privilege on the physical machines. You should actually not do this in practice! If you have your own laptop, by all means, install docker directly on your host OS.

❒ Go to the following page:`https://docs.docker.com/engine/installation/` and follow the instructions until you can run the `sudo docker run hello-world` and also create the following screen:

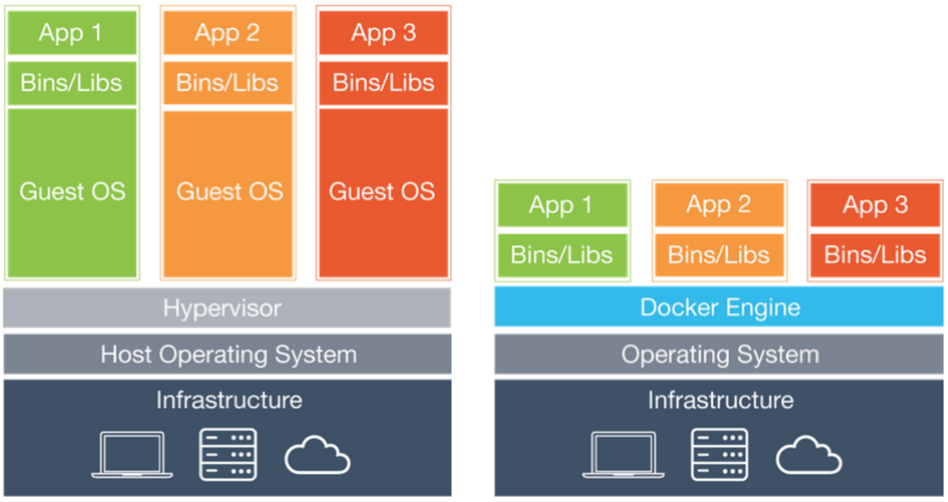
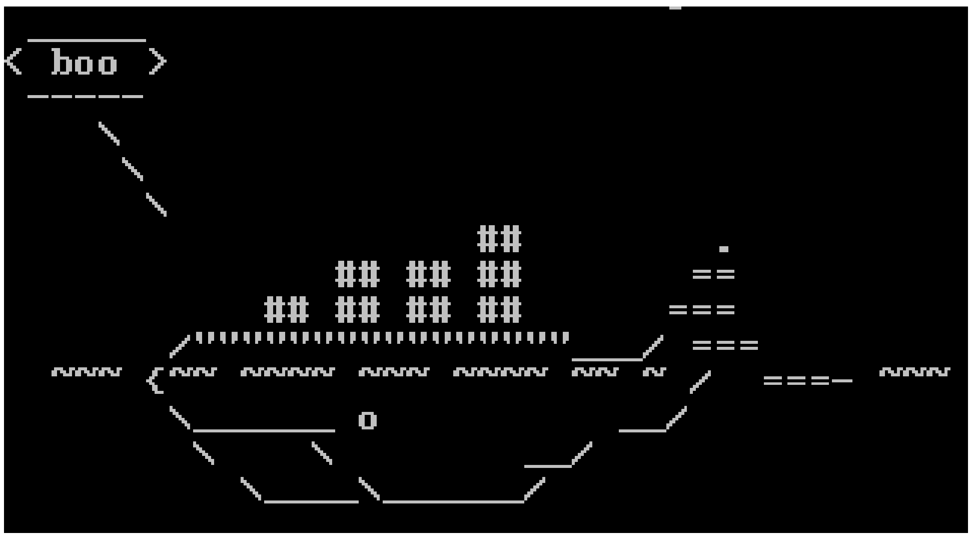


Figure 1: The above figure compares the two architectures. Note that in particular, although the docker engine creates a virtual platform for the apps to run on, it re-uses a lot of the resources of the host Kernel, as opposed to creating a separate guest Kernel per each app.



```
 _____
< boo >
 -----
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \    \        __/
          \____\______/
```