SauceLabs Coding Challenge

[Starts at 20:30PM PDT, 170719 ~ Ends at 20:30PM PDT, 190719]

- Journal -

Candidate

- **Han S. (Jay) Jang**
  jayjang.ca@gmail.com

Github

- https://github.com/jay-bcit/SauceLabs_CC.git

Journal

- https://docs.google.com/document/d/1DOC97dr9FoYRbR4xWZPS37w8tzWVJ73GDb1a-Tmd7zY/edit?usp=sharing

[Pre-requirements]

1. **Automation sources**

    - Virtualbox
        - Linux : **ubuntu-18.04.2**
            - Installer ISO : **ubuntu-18.04.2-server-amd64.iso**

2. **Requiremented packages**

    [Hostmachine]
    - Packer
    - Oracle Virtualbox
    - QEMU/KVM
    - Vagrant
    - Vncviewer
    - Inspec

[Things To Do]

1. **Configuration of VM image**

   [MARKS]
   1. **Time Zone** on the machine set to **UTC**
   2. **Enabling RDP** access if Windows, or **SSH** if **Linux**/**Mac**
      - (enabling RDP on Windows requires at least Pro or a Server version)
      - Bonus points for **VNC**
   3. **Enabling** firewall **ports RDP** or **SSH ports** in the OS if necessary, without actually disabling the firewall if it is already enabled
   4. **Disabling** automatic updates if **Windows** or Mac
   5. **Creating** a new **admin user** and **home directory** if **Linux**
   6. BONUS points if the new **VM** also has **all** of its **system software** **up to date**

3. **Archiving the resultant**

   - VM archive : **.ovf**
   - Virtual Disk : **.vdi / .vmdk**
   - QEMU/KVM : **qcow2**

4. **Assertion of automation**

   [MARKS]
   1. This assertion automation will **output** some kind of **status**
      - (process/shell **exit code** for example) to indicate whether it was a **pass** or fail
   2. BONUS points if this also calls something else
      - (push notification, external service, **e-mail**, etc.) to **notify** you somehow of its status

[Tools Used]

1. **Operating System** with support to run Virtualization
   (Bonus points if you can do this on **Linux** with Qemu/**KVM)**

   - HostMachine : Host OS -> **CentOS7**

2. **Hypervisor Application**
   (If using Linux, **Ubuntu** is preferred)

   - QEMU/KVM : GuestOS -> **Ubuntu_18.04**
   - VirtualBox : GuestOS -> **Ubuntu_18.04**

3. **Automation**

   - Packer
     - Linux/
       - ubuntu_18.04.2_**vbox**.json : preseed.cfg
       - ubuntu_18.04.2_**vbox_w_vagrant**.json : preseed.cfg
       - ubuntu_18.04.2_**qemu**.json : preseed_**qemu**.cfg

4. **Provisioner**

   - Shell
     - scripts/
       - Init.sh : Add **sudoer** user
       - update.sh : **Packages** check & update, **Disable** daily updates
       - vnc.sh : **VNC** setup and **cronjob** for **@reboot**
       - ufw.sh : **Firewall** settings + **sshd** hardening
       - postfix.sh : **MTA** setup for automation test result **notification**
       - check_list.sh : **Automation** tests in **Bash** script
       - cleanup.sh : **OOBE** settings

5. **TextEditor**

   - Atom : https://atom.io/

[Tasks]

1. **Directory Tree**

```
[hjang@sw07 Linux]$ ls ~/SauceLabs_CC/Linux/ | tree
.
|-- http
|   |-- preseed.cfg
|   `-- preseed_qemu.cfg
|-- scripts
|   |-- check_list.sh
|   |-- cleanup.sh
|   |-- init.sh
|   |-- postfix.sh
|   |-- ufw.sh
|   |-- update.sh
|   `-- vnc.sh
|-- ubuntu_18.04.2_qemu.json
|-- ubuntu_18.04.2_qemu_w_vgrant.json
|-- ubuntu_18.04.2_vbox.json
|-- ubuntu_18.04.2_vbox_w_vagrant.json
`-- Vagrantfile

2 directories, 14 files
[hjang@sw07 Linux]$
```

Explanation

A. http/

- pressed.cfg
- pressed_qemu.cfg

=> A localhost **pressed**/**url** directory to serve pressed.cfg file when boot from ISO

-> The file name has to be matched in "**builders**" block in Packer **.json** file

-> Option "**http_directory**"

-> Provide a file in "**boot_command**"

```
            " grub-installer/bootdev=/dev/vda<wait>",
            " noapic<wait>",
            " preseed/url=http://{{ .HTTPIP }}:{{ .HTTPPort }}/preseed_qemu.cfg<wait>",
            " -- <wait>",
            "<enter><wait>"
        ],
        "boot_wait": "10s",
        "disk_size": "{{user `disk_size`}}",
        "headless": false,
        "http_directory": "{{user `http_directory`}}",
        "output_directory": "{{user `output_directory`}}",
```

```
    "builders": [
        {
            "type": "qemu",
            "accelerator": "kvm",
            "boot_command": [
                "<esc><wait>",
                "<esc><wait>",
                "<enter><wait>",
                "/install/vmlinuz<wait>",
                " auto<wait>",
                " console-setup/ask_detect=false<wait>",
                " console-setup/layoutcode=us<wait>",
                " console-setup/modelcode=pc105<wait>",
                " debconf/frontend=noninteractive<wait>",
                " debian-installer=en_US<wait>",
                " fb=false<wait>",
                " initrd=/install/initrd.gz<wait>",
                " kbd-chooser/method=us<wait>",
                " keyboard-configuration/layout=USA<wait>",
                " keyboard-configuration/variant=USA<wait>",
                " locale=en_US<wait>",
                " netcfg/get_domain={{user `get_domain`}}<wait>",
                " netcfg/get_hostname={{user `get_hostname`}}<wait>",
                " grub-installer/bootdev=/dev/vda<wait>",
                " noapic<wait>",
                " preseed/url=http://{{ .HTTPIP }}:{{ .HTTPPort }}/preseed_qemu.cfg<wait>",
                " -- <wait>",
                "<enter><wait>"
            ],
```

[Reference]

Packer > Builder> QEMU Builder
https://www.packer.io/docs/builders/qemu.html

You must add a valid kickstart file to the "http_directory" and then provide the file in the "boot_command" in order for this build to run. We recommend you check out the Community Templates for a practical usage example.

Boot Command

https://www.packer.io/docs/builders/qemu.html#boot-command

The `boot_command` configuration is very important: it specifies the keys to type when the virtual machine is first booted in order to start the OS installer. This command is typed after `boot_wait`, which gives the virtual machine some time to actually load the ISO.

As documented above, the `boot_command` is an array of strings. The strings are all typed in sequence. It is an array only to improve readability within the template.

The boot command is "typed" character for character over a VNC connection to the machine, simulating a human actually typing the keyboard.

[Reference]

Provisioning a Development Environment with Packer, Part 1
https://www.endpoint.com/blog/2014/03/12/provisioning-development-environment

Installing your OS

The boot_command is a series of keystrokes that you can send to the machine via VNC. If you have setup a linux machine from scratch you know that you have to enter in a bunch of information to the machine about how to set it up for the first time such as time zone, keyboard layout, how to partition the hard drive, host name, etc. All these keystrokes needed to setup your machine can be used here. But if you think about it, that's a ton of keystrokes and this command could get quite long. A better way to approach this is to use a preseed file. A preseed.cfg file contains the same information you enter when you setup a machine for the first time. This isn't something provided by packer, but it is provided by the operating system to automatically provision machines. For **Ubuntu**, a preseed file is used like so:

- When you **boot from the startup media** (in this case an iso), you can choose the location of the **preseed** file via a url
- The **preseed** file is uploaded into memory and the configuration is read
- The installation process begins using information from the preseed file to enter the values where the user would normally enter them.

So how do we get the preseed file up to the machine? Remember that little web server that packer sets up? Well, the ip and port is made available to the virtual machine when it boots from the ISO. The following line tells the OS where to find the web server and the configuration file:

```
"preseed/url=http://{{ .HTTPIP }}:{{ .HTTPPort }}/precise_preseed.cfg"
```

B. scripts/

=> Directory that contains **Bash scripts** written in **"provisioners"** block in **.json** file

-> **Relative** Path

-> Runs as **root** inside GuestOS after installation process to configure a VM

-> Option "**scripts**"

```
"provisioners": [{
  "type": "shell",
  "execute_command": "echo '{{user `sudoer_name`}}' | {{.Vars}} sudo -S -E bash '{{.Path}}'",
  "scripts": [
    "scripts/init.sh",
    "scripts/update.sh",
    "scripts/vnc.sh",
    "scripts/ufw.sh",
    "scripts/postfix.sh",
    "scripts/check_list.sh",
    "scripts/cleanup.sh"
  ]
}]
```

[Reference]

Packer > Provisioner > Shell Provisioner
https://www.packer.io/docs/provisioners/shell.html#configuration-reference

`scripts` (array of strings) - An array of scripts to execute. The scripts will be uploaded and executed in the order specified. Each script is executed in isolation, so state such as variables from one script won't carry on to the next.

C. .json

=> Template that written in **JSON** so **Packer** uses
-> **Template Structure** in Packer

 # Example: ubuntu_18.04.2_**qemu**_w_vgrant.json
(Click the **link** to see the details of each option)

-> Variables

```
"variables": {
  "get_domain": "saucelabs03.test",
  "get_hostname": "ubuntu03",
  "disk_size": "8192",
  "guest_os_type": "Ubuntu_64",
  "http_directory": "http",
  "iso_checksum_type": "file",
  "iso_checksum_url": "http://cdimage.ubuntu.com/ubuntu/releases/bionic/release/SHA256SUMS",
  "iso_urls": "http://cdimage.ubuntu.com/ubuntu/releases/bionic/release/ubuntu-18.04.2-server-amd64.iso",
  "ssh_username": "vagrant",
  "ssh_password": "vagrant",
  "sudoer_name": "vagrant",
  "vm_name": "packer-ubuntu-18.04.2-amd64-qemu",
  "output_directory": "output-virtualbox-qemu"

},
  "sensitive-variables": ["ssh_username", "ssh_password"],
```


"**variables**": {
  "get_domain": "saucelabs03.test",
  "get_hostname": "ubuntu03",
  "disk_size": "8192",
  "guest_os_type": "Ubuntu_64",
  "http_directory": "http",
  "iso_checksum_type": "file",
  "iso_checksum_url":
"http://cdimage.ubuntu.com/ubuntu/releases/bionic/release/SHA256SUMS",
  "iso_urls":
"http://cdimage.ubuntu.com/ubuntu/releases/bionic/release/ubuntu-18.04.2-server-amd64.iso",
  "ssh_username": "vagrant",
  "ssh_password": "vagrant",
  "sudoer_name": "vagrant",
  "vm_name": "packer-ubuntu-18.04.2-amd64-qemu",
  "output_directory": "output-virtualbox-qemu",

  "output_vg_box": "ubuntu-18.04.2-qemu.box"

},
  "**sensitive-variables**": ["ssh_username", "ssh_password"],

-> Builders

```
23    "builders": [
24        {
25            "type": "qemu",
26            "accelerator": "kvm",
27            "boot_command": [
28                "<esc><wait>",
29                "<esc><wait>",
30                "<enter><wait>",
31                "/install/vmlinuz<wait>",
32                " auto<wait>",
```

- All variables are declared in “**variables**” block

```
"builders": [
  {
    "type": "qemu",
    "accelerator": "kvm",
    "boot_command": [
      "<esc><wait>",
      "<esc><wait>",
      "<enter><wait>",
      "/install/vmlinuz<wait>",
      " auto<wait>",
      " console-setup/ask_detect=false<wait>",
      " console-setup/layoutcode=us<wait>",
      " console-setup/modelcode=pc105<wait>",
      " debconf/frontend=noninteractive<wait>",
      " debian-installer=en_US<wait>",
      " fb=false<wait>",
      " initrd=/install/initrd.gz<wait>",
      " kbd-chooser/method=us<wait>",
      " keyboard-configuration/layout=USA<wait>",
      " keyboard-configuration/variant=USA<wait>",
      " locale=en_US<wait>",
      " netcfg/get_domain={{user `get_domain`}}<wait>",
      " netcfg/get_hostname={{user `get_hostname`}}<wait>",
      " grub-installer/bootdev=/dev/vda<wait>",
      " noapic<wait>",
      " preseed/url=http://{{ .HTTPIP }}:{{ .HTTPPort }}/preseed_qemu.cfg<wait>",
```

```
        " -- <wait>",
        "<enter><wait>"
      ],
      "boot_wait": "10s",
      "disk_size": "{{user `disk_size`}}",
      "headless": false,
      "http_directory": "{{user `http_directory`}}",
      "output_directory": "{{user `output_directory`}}",
      "ssh_username": "{{user `ssh_username`}}",
      "ssh_password": "{{user `ssh_password`}}",
      "ssh_port": 22,
      "ssh_wait_timeout": "10000s",
      "iso_urls": "{{user `iso_urls`}}",
      "iso_checksum_type": "{{user `iso_checksum_type`}}",
      "iso_checksum_url": "{{user `iso_checksum_url`}}",
      "shutdown_command": "echo '{{user `sudoer_name`}}'|sudo -S shutdown -P now",
      "vm_name": "{{user `vm_name`}}",
      "net_device": "virtio-net",
      "disk_interface": "virtio",
      "format": "qcow2",
      "qemu_binary": "qemu-system-x86_64",
      "qemuargs": [
        [ "-m", "4092M" ],
        [ "-smp", "cpus=2,maxcpus=2,cores=2" ]
      ]
    }
  ],
```

->

```
      "format": "qcow2",                    : Output format of the virtual machine image.
      "qemu_binary": "qemu-system-x86_64",  : Has to be matched with system's binary
      "qemuargs": [                         : VM Hardware Configuration
```

-> Provisioners

```
"provisioners": [{
  "type": "shell",
  "execute_command": "echo '{{user `sudoer_name`}}' | {{.Vars}} sudo -S -E bash '{{.Path}}'",
  "scripts": [
    "scripts/init.sh",
    "scripts/update.sh",
    "scripts/vnc.sh",
    "scripts/ufw.sh",
    "scripts/postfix.sh",
    "scripts/check_list.sh",
    "scripts/cleanup.sh"
  ]
}]
```

- **Shell** has been used as **Provisioner**

```
"provisioners": [{
  "type": "shell",
  "execute_command": "echo '{{user `sudoer_name`}}' | {{.Vars}} sudo -S -E bash '{{.Path}}'",
  "scripts": [
    "scripts/init.sh",
    "scripts/update.sh",
    "scripts/vnc.sh",
    "scripts/ufw.sh",
    "scripts/postfix.sh",
    "scripts/check_list.sh",
    "scripts/cleanup.sh"
  ]
}],
```

[Reference]

Packer > Template Provisioners
https://www.packer.io/docs/templates/provisioners.html

Packer > Shell Provisioner
https://www.packer.io/docs/provisioners/shell.html

-> Post-Processors

```
"post-processors": [{
  "type": "vagrant",
  "output": "{{user `output_vg_box`}}"
}]
```

- **Vagrant** has been used as **Post-Processors**
    - Only exist in ***_w_vagrant.json** files

"**post-processors**": [{
  "**type**": "**vagrant**",
  "**output**": "{{user `output_vg_box`}}"
}]

[Reference]

Packer > Template Post-Processors
https://www.packer.io/docs/templates/post-processors.html

Packer > Vagrant Post-Processor
https://www.packer.io/docs/post-processors/vagrant.html

[Reference]

Template Structure

https://www.packer.io/docs/templates/index.html#template-structure

A template is a JSON object that has a set of keys configuring various components of Packer. The available keys within a template are listed below. Along with each key, it is noted whether it is required or not.

- `builders` (*required*) is an array of one or more objects that defines the builders that will be used to create machine images for this template, and configures each of those builders. For more information on how to define and configure a builder, read the sub-section on configuring builders in templates.
- `description` (optional) is a string providing a description of what the template does. This output is used only in the inspect command.
- `min_packer_version` (optional) is a string that has a minimum Packer version that is required to parse the template. This can be used to ensure that proper versions of Packer are used with the template. A max version can't be specified because Packer retains backwards compatibility with `packer fix`.
- `post-processors` (optional) is an array of one or more objects that defines the various post-processing steps to take with the built images. If not specified, then no post-processing will be done. For more information on what post-processors do and how they're defined, read the sub-section on configuring post-processors in templates.
- `provisioners` (optional) is an array of one or more objects that defines the provisioners that will be used to install and configure software for the machines created by each of the builders. If it is not specified, then no provisioners will be run. For more

information on how to define and configure a provisioner, read the sub-section on configuring provisioners in templates.

- `variables` (optional) is an object of one or more key/value strings that defines user variables contained in the template. If it is not specified, then no variables are defined. For more information on how to define and use user variables, read the sub-section on user variables in templates.

2. GuestOS Configuration

A. Preseed_Configuration

- **Kickstart** file for **Ubuntu** : Customized settings for OS installation

  - http/**preseed_qemu.cfg**
  - http/**preseed.cfg**

-> Key Point 1.

- **Disk path** for **QEMU/KVM**

```
#################################################################
## QEMU/KVM setting => /dev/vda ##

d-i partman-auto/disk string /dev/vda

#################################################################
```

- **Disk path** for **Vbox**

```
#################################################################
## Vbox setting => /dev/sda ##

d-i partman-auto/disk string /dev/sda

#################################################################
```

-> Key Point 2.

- **Custom user settings**

: vagrant

[MARKS] 5. **Creating** a new **admin user** and **home directory** if **Linux**

```
##########################################################
## Create user account  : vagrant ##

# To create a normal user account.
d-i passwd/user-fullname string vagrant
d-i passwd/username string vagrant

# Normal user's password, either in clear text
d-i passwd/user-password password vagrant
d-i passwd/user-password-again password vagrant

# The installer will warn about weak passwords. If you are sure you know
# what you're doing and want to override it, uncomment this.
d-i user-setup/allow-password-weak boolean true

# Set to true if you want to encrypt the first user's home directory.
d-i user-setup/encrypt-home boolean false

# The user account will be added to some standard initial groups. To override that, use this.
d-i passwd/user-default-groups vagrant sudo

# Create the first user with the specified UID instead of the default.
d-i passwd/user-uid string 900
##########################################################
```

-> Key Point 3.

- **Time zone Setting : UTC**

-> [MARKS] 1. **Time Zone** on the machine set to **UTC**

```
#################################################################
## time zone ##

d-i time/zone string UTC

#################################################################
```

[Reference]

Appendix B. Automating the installation using preseeding
https://help.ubuntu.com/lts/installation-guide/s390x/apb.html

B.4. Contents of the preconfiguration file (for bionic)
https://help.ubuntu.com/lts/installation-guide/s390x/apbs04.html

Contents of the preconfiguration file (for stretch)
https://help.ubuntu.com/lts/installation-guide/example-preseed.txt

no root file system on ubuntu 16 packer install
https://askubuntu.com/questions/831887/no-root-file-system-on-ubuntu-16-packer-install

Packer – automating virtual machine image creation
http://alexconst.net/2016/01/11/packer/

B. Run Scripts

- Packer uploads scripts that declared in **.json** template file
- and **run** its as **root** inside GuestOS => **Shell** provisioner
- **Relative Path**

```
"provisioners": [{
  "type": "shell",
  "execute_command": "echo '{{user `sudoer_name`}}' | {{.Vars}} sudo -S -E bash '{{.Path}}'",
  "scripts": [
    "scripts/init.sh",
    "scripts/update.sh",
    "scripts/vnc.sh",
    "scripts/ufw.sh",
    "scripts/postfix.sh",
    "scripts/check_list.sh",
    "scripts/cleanup.sh"
  ]
}]
```

[Reference]

Packer > Configuration Reference
https://www.packer.io/docs/provisioners/shell.html#scripts

 # Scripts lists

1. Init.sh

- **Add User** to **sudoer** group
- **Username** can be modified

 # Variables

declare user_name=**vagrant**

```
#!/bin/bash -x
set -o nounset  # Treat unset variables as an error

declare user_name=vagrant

# Add vagrant user to sudoers
echo "$user_name        ALL=(ALL)       NOPASSWD: ALL" >> /etc/sudoers
sed -i "s/^.*requiretty/#Defaults requiretty/" /etc/sudoers

```

[Reference]

Linux Tips: Password Usage in Sudo (PASSWD / NOPASSWD)
http://www.ducea.com/2006/06/18/linux-tips-password-usage-in-sudo-passwd-nopasswd/

2. update.sh

- **Update packages** and **upgrades** it
- **Disable daily** apt unattended **updates**
- Create each package list at the moment to compare to see what packages has been added / changed
  -> Both files will be used in **check_list.sh**

[MARKS] 6. BONUS points if the new **VM** also has **all** of its **system software up to date**

```
#!/bin/bash -xeu
set -o nounset  # Treat unset variables as an error

# Record current package lists
dpkg -l > packge_list_01.txt
echo "Last update: $(date -R)" >> packge_list_01.txt

# Check packages that needs to be updated and upgrade it
sudo apt update -y
sudo apt upgrade -y

# Record package lists after update to compare in check_list.sh
dpkg -l > packge_list_02.txt
echo "Last update: $(date -R)" >> packge_list_02.txt

# Disable daily apt unattended updates
echo 'APT::Periodic::Enable "0";' >> /etc/apt/apt.conf.d/10periodic
```

[Reference]

How to Enable/Disable Unattended Upgrades in Ubuntu 16.04
https://linoxide.com/ubuntu-how-to/enable-disable-unattended-upgrades-ubuntu-16-04/

How to upgrade to Ubuntu Linux 18.04
https://www.zdnet.com/article/how-to-upgrade-to-ubuntu-linux-18-04/

3. vnc.sh

- **Install vnc4server** package + **xfce4** GUI
- **Set** default **emulator** as **xfce4-terminal.wrapper**
- Add a **subscript** to assure idempotency (Run service **@reboot**)

-> /etc/cron.d/cron_vnc

[MARKS] 2. **Enabling RDP** access if Windows, or **SSH** if **Linux**/**Mac**

- (enabling RDP on Windows requires at least Pro or a Server version)
- Bonus points for **VNC**

```bash
#!bin/bash -x
set -o nounset  # Treat unset variables as an error

declare vnc_pass="P@ssw0rd"
declare vnc_emulator="/etc/alternatives/x-terminal-emulator"
declare target_emulator="/usr/bin/xfce4-terminal.wrapper"

# install vncserver, xfce desktop manager
sudo apt-get install vnc4server xfce4 xfce4-goodies -y


# Change the symlink to default emulator as xfce4-terminal.wrapper
if [[ -s $vnc_emulator ]]; then
  if [[ $(ll $vnc_emulator | cut -d " " -f 11) != "$target_emulator*" ]]; then
    sudo unlink $vnc_emulator && sudo ln -s $target_emulator $vnc_emulator
    echo "[SYSTEM] - CHANGED - Current emuloator for VNC : $(ll $vnc_emulator | cut -d " " -f 11)"
  else
    echo "[SYSTEM] - NO_CHANGED - Current emuloator for VNC : $(ll $vnc_emulator | cut -d " " -f 11)"
  fi
else
  echo "[SYSTEM] - NULL - Current emulator has no symlink to the wrapper"
fi


# Configure VNC password
sudo umask 0077                                          # use safe default permissions
sudo mkdir -p "/root/.vnc"                               # create config directory
sudo chmod go-rwx "/root/.vnc"                           # enforce safe permissions
sudo printf "$vnc_pass\n$vnc_pass\n\n" | sudo vncpasswd  # generate and write a password
#vncpasswd -f <<<"$vnc_pass" >"$HOME/.vnc/passwd"
```

```
# Create xstartup file
sudo cat <<EOF > /root/.vnc/xstartup
#!/bin/bash
startxfce4 &
EOF
sudo chmod +x /root/.vnc/xstartup


# Create script for crontab to start vnc4server at boot
sudo umask 0077                                                # use safe default permissions
sudo mkdir -p "/root/crontabs"                                 # create config directory
sudo chmod go-rwx "/root/crontabs"                             # enforce safe permissions
sudo cat <<EOF > /root/crontabs/cron_vnc.sh
#!/bin/bash
sudo printf "$vnc_pass\n$vnc_pass\n\n" | sudo /usr/bin/vnc4server
EOF
sudo chmod +x /root/crontabs/cron_vnc.sh

# Add script to crontabs
sudo echo "@reboot root /root/crontabs/cron_vnc.sh" > /etc/cron.d/cron_vnc




# Run vncserver
sudo vnc4server
```

[Reference]

VNC server on Ubuntu 18.04 Bionic Beaver Linux
https://linuxconfig.org/vnc-server-on-ubuntu-18-04-bionic-beaver-linux

CentOS > VNC (Virtual Network Computing)
https://wiki.centos.org/HowTos/VNC-Server

vncpasswd(1) Manual Page
https://www.tightvnc.com/vncpasswd.1.php

Set up TightVNC programmatically with BASH
https://stackoverflow.com/questions/30606655/set-up-tightvnc-programmatically-with-bash

How to cat <<EOF >> a file containing code?
https://stackoverflow.com/questions/22697688/how-to-cat-eof-a-file-containing-code/22698106

Starting VNC Server on Boot
https://www.linode.com/docs/applications/remote-desktop/install-vnc-on-ubuntu-16-04/#starting-vnc-server-on-boot

crontab's @reboot only works for root?
https://unix.stackexchange.com/questions/109804/crontabs-reboot-only-works-for-root

Location of users cron files
https://www.unix.com/unix-for-dummies-questions-and-answers/16374-location-users-cron-files.html

Does Ubuntu support “@reboot” in crontab?
https://askubuntu.com/questions/335615/does-ubuntu-support-reboot-in-crontab

@reboot - explaining simple cron magic
https://www.unixdaemon.net/linux/how-does-cron-reboot-work/

Run a script only at the very first boot
https://askubuntu.com/questions/156771/run-a-script-only-at-the-very-first-boot

How To Install and Configure VNC Remote Access for the GNOME Desktop on CentOS 7
https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-vnc-remote-access-for-the-gnome-desktop-on-centos-7

How to Install and Configure VNC on Ubuntu 16.04
https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-vnc-on-ubuntu-16-04

Redhat > 13.3. VNC VIEWER
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/sec-vnc-viewer

Connecting to VNC Server via VNC Client
https://www.tecmint.com/install-and-configure-vnc-server-on-ubuntu/

Getting input/output error running anything on remote desktop
https://askubuntu.com/questions/899391/getting-input-output-error-running-anything-on-remote-desktop

update-alternatives(8) - Linux man page
https://linux.die.net/man/8/update-alternatives

How to Change or Remove Symbolic Link in Linux?
https://www.hostinger.com/tutorials/how-to-create-symbolic-links-in-linux/

4.ufw.sh

- **Firewall settings** for **services**
  - sshd : Port22
  - vnc4server : Port5901

- **Disable IPv6** in **sshd** (sshd hardening)

```
#!bin/bash -x
set -o nounset  # Treat unset variables as an error

# Firewall (ufw) configuration
## 1. Reset to defualt and enable ufw
sudo ufw --force reset
sudo ufw --force enable

## 2. Add rules: base_line
sudo ufw default deny incoming
sudo ufw default allow outgoing

## 3. Add ruels: sshd
sudo ufw allow ssh

## 4. Add ruels: vnc4server
sudo ufw allow from any to any port 5901 proto tcp


# Hardening sshd - simple_ver
## 1. Disable IPv6 for sshd + Backup original config file
cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bck
sed -i "s/#ListenAddress 0.0.0.0/ListenAddress 0.0.0.0/" /etc/ssh/sshd_config

## 2. Restart service: sshd
sudo systemctl restart sshd

```

[Reference]

How To Set Up a Firewall with UFW on Ubuntu 14.04
https://www.digitalocean.com/community/tutorials/how-to-set-up-a-firewall-with-ufw-on-ubuntu-14-04

Disable IPv6 on Ubuntu 12.04
https://easyengine.io/tutorials/linux/disable-ipv6/

5.postfix.sh

- **MTA** setup for **e-mail notification**
  - Will be used in **check_list.sh**
  - **<u>Keep it as simple</u>**

```bash
#!bin/bash -x
set -o nounset  # Treat unset variables as an error

declare mail_hostname=$(hostname -f)

# INstall postfix
sudo debconf-set-selections <<< "postfix postfix/mailname string $mail_hostname"
sudo debconf-set-selections <<< "postfix postfix/main_mailer_type string 'Internet Site'"
sudo apt-get install mailutils -y


# Set postfix as send-only server
#sed -i "s/inet_interfaces = all/inet_interfaces = loopback-only/" /etc/postfix/main.cf
echo << EOF >> /etc/postfix/main.cf
inet_interfaces = loopback-only
myhostname=$mail_hostname
EOF

# Reload modified configuration
sudo systemctl restart postfix

# Enable postfix
sudo systemctl enable postfix

```

[Reference]

How to Setup Postfix as Send-only Mail Server on an Ubuntu 18.04 Dedicated Server or VPS
https://hostadvice.com/how-to/how-to-setup-postfix-as-send-only-mail-server-on-an-ubuntu-18-04-dedicated-server-or-vps/

Automate the installation of postfix on Ubuntu
https://serverfault.com/questions/143968/automate-the-installation-of-postfix-on-ubuntu

5 Ways to Send Email From Linux Command Line
https://tecadmin.net/ways-to-send-email-from-linux-command-line/

mail: cannot send message: process exited with a non-zero status
https://unix.stackexchange.com/questions/185365/mail-cannot-send-message-process-exited-with-a-non-zero-status

[SOLVED] Sendmail / Apache
https://bbs.archlinux.org/viewtopic.php?id=147428

Sending simple message body + file attachment using Linux Mailx [duplicate]
https://stackoverflow.com/questions/8314999/sending-simple-message-body-file-attachment-using-linux-mailx

6.check_list.sh

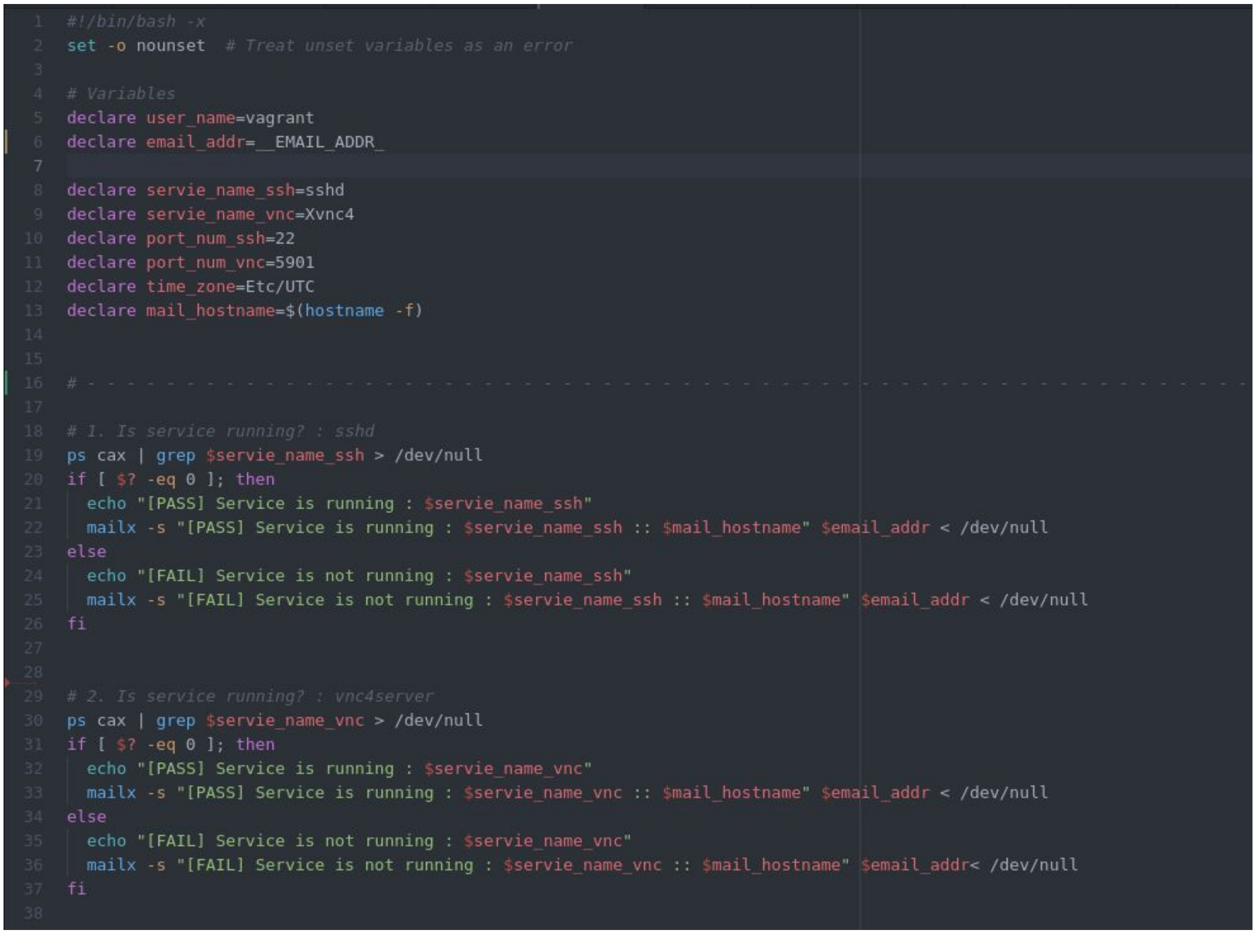
- **Automation assertion test** script
    - Each test triggers ***mailx*** command to send an email with its own results

    - # 1. Is **service** running? : sshd
    - # 2. Is **service** running? : vnc4server
    - # 3. Is **port** opend? : sshd
    - # 4. Is **port** opend? : vnc4server
    - # 5. New **user** added and **sudoers**?
    - # 6. Does **user** has **home directory**?
    - # 7. Is **timezone** set by **UTC**?
    - # 8. Compare package lists to see **system software** is **up to date**

[MARKS] 1. This assertion automation will **output** some kind of **status**
[MARKS] 2. <u>BONUS</u> points if this also calls something else
- (push notification, external service, **e-mail**, etc.) to **notify** you somehow of its status

Variables

declare user_name=**vagrant**
declare email_addr=**__EMAIL_ADDR_**



```bash
#!/bin/bash -x
set -o nounset  # Treat unset variables as an error

# Variables
declare user_name=vagrant
declare email_addr=__EMAIL_ADDR_

declare servie_name_ssh=sshd
declare servie_name_vnc=Xvnc4
declare port_num_ssh=22
declare port_num_vnc=5901
declare time_zone=Etc/UTC
declare mail_hostname=$(hostname -f)


# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# 1. Is service running? : sshd
ps cax | grep $servie_name_ssh > /dev/null
if [ $? -eq 0 ]; then
  echo "[PASS] Service is running : $servie_name_ssh"
  mailx -s "[PASS] Service is running : $servie_name_ssh :: $mail_hostname" $email_addr < /dev/null
else
  echo "[FAIL] Service is not running : $servie_name_ssh"
  mailx -s "[FAIL] Service is not running : $servie_name_ssh :: $mail_hostname" $email_addr < /dev/null
fi


# 2. Is service running? : vnc4server
ps cax | grep $servie_name_vnc > /dev/null
if [ $? -eq 0 ]; then
  echo "[PASS] Service is running : $servie_name_vnc"
  mailx -s "[PASS] Service is running : $servie_name_vnc :: $mail_hostname" $email_addr < /dev/null
else
  echo "[FAIL] Service is not running : $servie_name_vnc"
  mailx -s "[FAIL] Service is not running : $servie_name_vnc :: $mail_hostname" $email_addr< /dev/null
fi

```

```
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -




# 3. Is port opend? : sshd
ss -lnt | grep $port_num_ssh > /dev/null
if [ $? -eq 0 ]; then
  echo "[PASS] Port opend and listning: $port_num_ssh"
  mailx -s "[PASS] Port opend and listning: $port_num_ssh :: $mail_hostname" $email_addr < /dev/null
else
  echo "[FAIL] Port is not opend : $port_num_ssh"
  mailx -s "[FAIL] Port is not opend : $port_num_ssh :: $mail_hostname" $email_addr < /dev/null
fi



# 4. Is port opend? : vnc4server
ss -lnt | grep $port_num_vnc > /dev/null
if [ $? -eq 0 ]; then
  echo "[PASS] Port opend and listning: $port_num_vnc"
  mailx -s "[PASS] Port opend and listning: $port_num_vnc :: $mail_hostname" $email_addr < /dev/null
else
  echo "[FAIL] Port is not opend: $port_num_vnc"
  mailx -s "[FAIL] Port is not opend : $port_num_vnc :: $mail_hostname" $email_addr < /dev/null
fi




# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

```

```

# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - #




# 5. New user added and sudoers?
cat /etc/passwd | grep $user_name > /dev/null && sudo cat /etc/sudoers | grep $user_name > /dev/null
if [ $? -eq 0 ]; then
  echo "[PASS] User exist and has been added to sudoers: $user_name"
  mailx -s "[PASS] User exist and has been added to sudoers: $user_name :: $mail_hostname" $email_addr < /dev/null
else
  echo "[FAIL] User does not exist"
  mailx -s "[FAIL] User does not exist: $user_name :: $mail_hostname" $email_addr < /dev/null
fi




# 6. Does user has home directory?
ls /home | grep $user_name > /dev/null
if [ $? -eq 0 ]; then
  echo "[PASS] User home directory exists: Path => $(pwd /home/$user_name)"
  mailx -s "[PASS] User home directory exists: Path => $(pwd /home/$user_name) :: $mail_hostname" $email_addr < /dev/null
else
  echo "[FAIL] User home directory does not exist"
  mailx -s "[FAIL] User home directory does not exist: Username => $user_name :: $mail_hostname" $email_addr < /dev/null
fi


# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - #


```

```bash

# 7. Is timezone set by UTC?
if [[ $(cat /etc/timezone) && $(timedatectl | grep "Time zone" | cut -d " " -f 19) == "$time_zone" ]]; then
  if [ $? -eq 0 ]; then
    echo "[PASS] Timezone has been set by : $time_zone"
    mailx -s "[PASS] Timezone has been set by : $time_zone :: $mail_hostname" $email_addr < /dev/null
  else
    echo "[FAIL] Timezone has not been set"
    mailx -s "[FAIL] Timezone has not been set : Current Time_Zone:  $(timedatectl | grep "Time zone" | cut -d " " -f 19) :: $mail_hostname" $email_addr < /dev/null
  fi
fi



# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - #




# 8. Compare package lists to see system software is up to date
diff packge_list_01.txt packge_list_02.txt | grep "> ii" > /dev/null
if [ $? -eq 0 ]; then
  echo "[SYSTEM] List of packages that has been added / updated
$(diff packge_list_01.txt packge_list_02.txt)"
  diff packge_list_01.txt packge_list_02.txt > result.txt
  mailx -s "[SYSTEM] List of packages that has been added / updated :: $mail_hostname" $email_addr < result.txt
else
  echo "[SYSTEM] No packages has been changed"
  mailx -s "[SYSTEM] No packages has been changed :: $mail_hostname" $email_addr < /dev/null

fi

```

[Reference]

jay-bcit/project-script
https://bitbucket.org/jay-bcit/project-script/src/master/main.sh

Check if service exists in bash (CentOS and Ubuntu)
https://stackoverflow.com/questions/24398242/check-if-service-exists-in-bash-centos-and-ubuntu

how to test if the apt cache is up to date with bash
https://askubuntu.com/questions/487558/how-to-test-if-the-apt-cache-is-up-to-date-with-bash

Ubuntu 'apt-get' list of commands (cheat sheet)
https://alvinalexander.com/linux-unix/ubuntu-apt-get-cache-list-search-commands-cheat-sheet

Bash if condition to check if a ubuntu package has a newer version?
https://stackoverflow.com/questions/36455218/bash-if-condition-to-check-if-a-ubuntu-package-has-a-newer-version

How to Check and Set Timezone in Ubuntu
https://www.youtube.com/watch?v=i_m90hbvwwM

How does nested if/then/elseif work in bash? [closed]
https://stackoverflow.com/questions/15327973/how-does-nested-if-then-elseif-work-in-bash

How to represent multiple conditions in a shell if statement?
https://stackoverflow.com/questions/3826425/how-to-represent-multiple-conditions-in-a-shell-if-statement/3826462

Using the && operator in an if statement
https://stackoverflow.com/questions/16396146/using-the-operator-in-an-if-statement

send mail script only if condition is met
https://www.unix.com/unix-for-advanced-and-expert-users/168670-send-mail-script-only-if-condition-met.html

Bash - if statement combined with mail command
https://stackoverflow.com/questions/21799136/bash-if-statement-combined-with-mail-command

How to check if a file is empty in Bash?
https://stackoverflow.com/questions/9964823/how-to-check-if-a-file-is-empty-in-bash

7.cleanup.sh

- **Removes dependencies** from **uninstalled** packages.
- Delete **uploaded scripts** and **files** **(*.sh, *.txt)**
- **Zero out** the rest of the free space
- Sync: Makes **Packer** doesn't quit too early, before the large file is deleted

```
#!/bin/bash -xeu
set -o nounset  # Treat unset variables as an error

# Apt cleanup.: removes dependencies from uninstalled applications.
apt autoremove
apt update

# Delete unneeded files.
rm -f /home/vagrant/*.sh
rm -f /home/vagrant/*.txt

# Zero out the rest of the free space using dd, then delete the written file.
dd if=/dev/zero of=/EMPTY bs=1M
rm -f /EMPTY

# Add `sync` so Packer doesn't quit too early, before the large file is deleted.
sync

```

3. **Packer Build**

- In order to build new OS image, run command as below

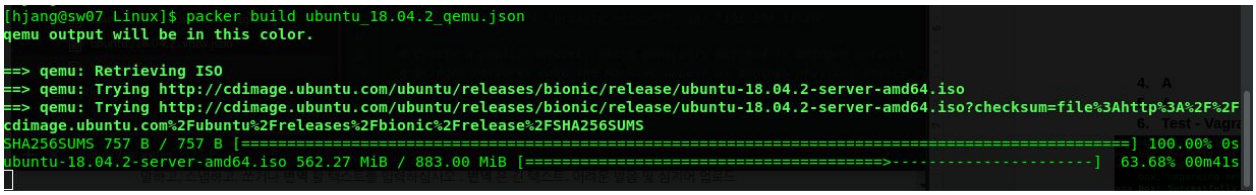
packer build <FILE_PATH>

- **File choices**
  - ubuntu_18.04.2_**qemu**.json : QEMU/KVM (**qcow2**)
  - ubuntu_18.04.2_vbox.json : VirtualBox (**.ovf, .vdi**)
  - ubuntu_18.04.2_**qemu**_w_**vgrant**.json : Vagrant (**.box**)
  - ubuntu_18.04.2_vbox_w_vagrant.json : Vagrant (**.box**)

Build Process

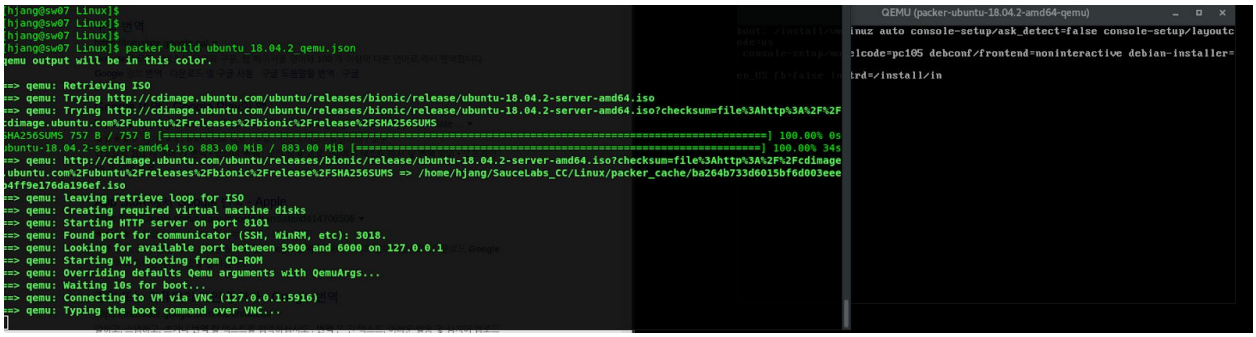
-> Packer downloads **ISO** image from mirror that written in **.json** template

-> WIll compare with **checksum** that has been downloaded

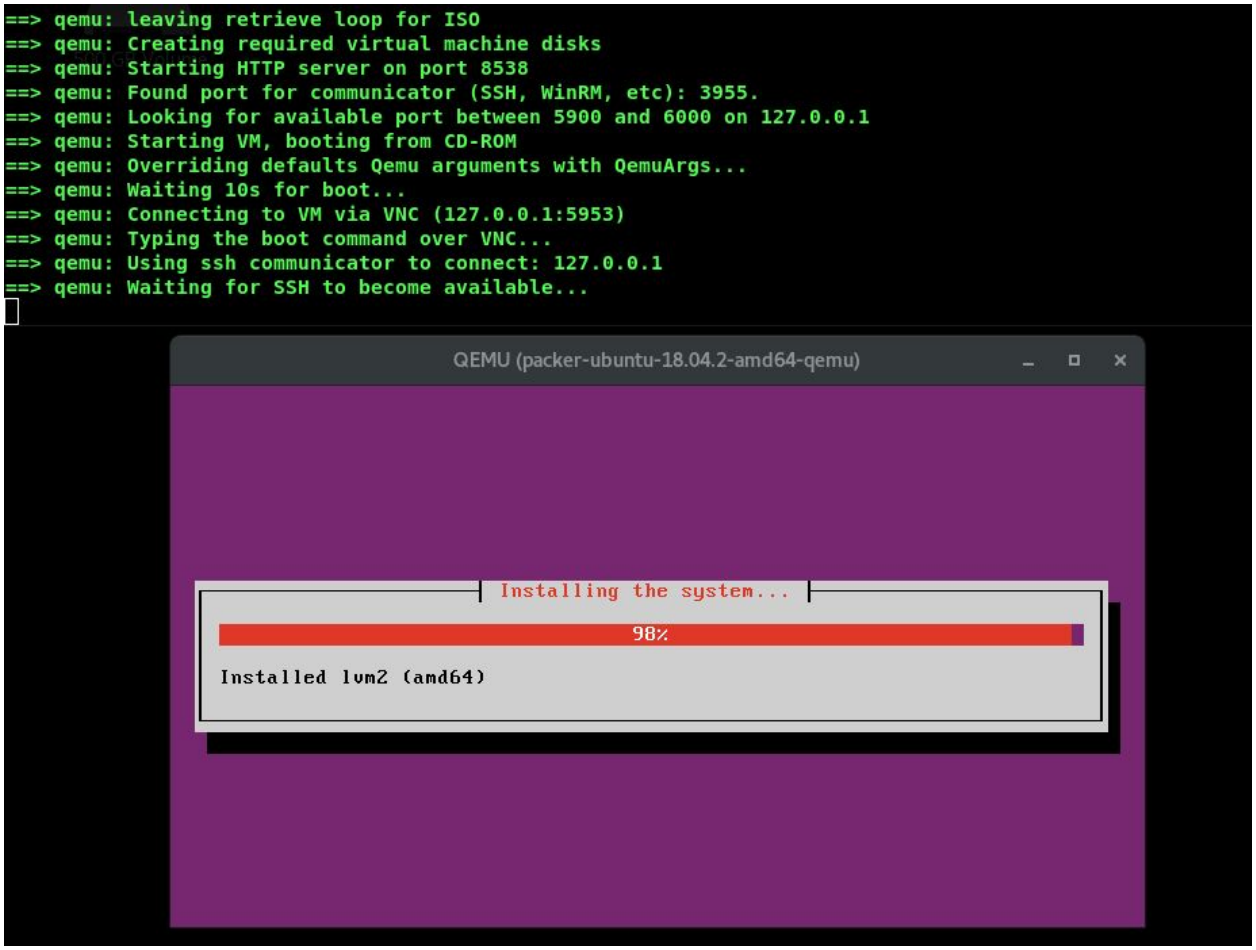


-> Packer types GuestOS installation command through **VNC**

-> **“builder”** block in **.json** template



-> Fully Automated **Guest OS installation** with **preeseed_qemu.cfg**



Provisioning Process

-> **Packer** uses **ssh** to **upload** / **run scripts** after system boot

-> **scripts/*.sh** has been uploaded and running in order

- init.sh

==> qemu: Provisioning with shell script: **scripts/init.sh**

- update.sh

==> qemu: Provisioning with shell script: **scripts/update.sh**

- vnc.sh

==> qemu: Provisioning with shell script: **scripts/vnc.sh**

```
==> qemu: Provisioning with shell script: scripts/vnc.sh
    qemu: Reading package lists...
    qemu: Building dependency tree...
    qemu: Reading state information...
    qemu: The following additional packages will be installed:
    qemu:   adwaita-icon-theme apport apport-symptoms aspell aspell-en at-spi2-core
    qemu:   dbus-user-session dbus-x11 dconf-gsettings-backend dconf-service
    qemu:   desktop-base desktop-file-utils dictionaries-common emacsen-common enchant
    qemu:   exo-utils fontconfig fontconfig-config fonts-dejavu-core gconf-service
    qemu:   gconf-service-backend gconf2 gconf2-common gdisk glib-networking
    qemu:   glib-networking-common glib-networking-services gnome-terminal
    qemu:   gnome-terminal-data greybird-gtk-theme gsettings-desktop-schemas
    qemu:   gstreamer1.0-gl gstreamer1.0-plugins-base gstreamer1.0-plugins-good
    qemu:   gstreamer1.0-pulseaudio gstreamer1.0-x gtk-update-icon-cache
    qemu:   gtk2-engines-murrine gtk2-engines-xfce gvfs gvfs-common gvfs-daemons
    qemu:   gvfs-libs hddtemp hicolor-icon-theme humanity-icon-theme hunspell-en-us
    qemu:   libaa1 libart-2.0-2 libasound2 libasound2-data libasound2-plugins
    qemu:   libaspell15 libasyncns0 libatasmart4 libatk-bridge2.0-0 libatk1.0-0
    qemu:   libatk1.0-data libatkmm-1.6-1v5 libatspi2.0-0 libauthen-sasl-perl
    qemu:   libavahi-client3 libavahi-common-data libavahi-common3 libavahi-glib1
    qemu:   libavc1394-0 libblockdev-crypto2 libblockdev-fs2 libblockdev-loop2
    qemu:   libblockdev-part-err2 libblockdev-part2 libblockdev-swap2 libblockdev-utils2
    qemu:   libblockdev2 libbonobo2-0 libbonobo2-common libbonoboui2-0
    qemu:   libbonoboui2-common libbrotli1 libburn4 libcaca0 libcairo-gobject2 libcairo2
    qemu:   libcairomm-1.0-1v5 libcanberra-gtk3-0 libcanberra-gtk3-module libcanberra0
```

-> Result of '**if**' statement in **vnc.sh**

```
    qemu: aspell-autobuildhash: processing: en [en_GB-variant_1].
    qemu: aspell-autobuildhash: processing: en [en_US-w_accents-only].
    qemu: aspell-autobuildhash: processing: en [en_US-wo_accents-only].
    qemu: Processing triggers for dbus (1.12.2-1ubuntu1.1) ...
    qemu: Processing triggers for libgdk-pixbuf2.0-0:amd64 (2.36.11-2) ...
    qemu: Processing triggers for initramfs-tools (0.130ubuntu3.8) ...
    qemu: update-initramfs: Generating /boot/initrd.img-4.15.0-54-generic
    qemu: [SYSTEM] - CHANGED - Current emuloator for VNC : /etc/alternatives/x-terminal-emulator
```

```
# Change the symlink to default emulator as xfce4-terminal.wrapper
if [[ -s $vnc_emulator ]]; then
  if [[ $(ls $vnc_emulator | cut -d " " -f 11) != "$target_emulator*" ]]; then
    sudo unlink $vnc_emulator && sudo ln -s $target_emulator $vnc_emulator
    echo "[SYSTEM] - CHANGED - Current emuloator for VNC : $(ls $vnc_emulator | cut -d " " -f 11)"
  else
    echo "[SYSTEM] - NO_CHANGED - Current emuloator for VNC : $(ls $vnc_emulator | cut -d " " -f 11)"
  fi
else
  echo "[SYSTEM] - NULL - Current emulator has no symlink to the wrapper"
fi
```

- ufw.sh

==> qemu: Provisioning with shell script: **scripts/ufw.sh**

[MARKS] 3. **Enabling** firewall **ports RDP** or **SSH ports** in the OS if necessary, without actually disabling the firewall if it is already enabled

```
==> qemu: Provisioning with shell script: scripts/ufw.sh
    qemu: Backing up 'user.rules' to '/etc/ufw/user.rules.20190719_073010'
    qemu: Backing up 'before.rules' to '/etc/ufw/before.rules.20190719_073010'
    qemu: Backing up 'after.rules' to '/etc/ufw/after.rules.20190719_073010'
    qemu: Backing up 'user6.rules' to '/etc/ufw/user6.rules.20190719_073010'
    qemu: Backing up 'before6.rules' to '/etc/ufw/before6.rules.20190719_073010'
    qemu: Backing up 'after6.rules' to '/etc/ufw/after6.rules.20190719_073010'
    qemu:
    qemu: Firewall is active and enabled on system startup
    qemu: Default incoming policy changed to 'deny'
    qemu: (be sure to update your rules accordingly)
    qemu: Default outgoing policy changed to 'allow'
    qemu: (be sure to update your rules accordingly)
    qemu: Rule added
    qemu: Rule added (v6)
    qemu: Rule added
    qemu: Rule added (v6)
```

- postfix.sh

==> qemu: Provisioning with shell script: **scripts/postfix.sh**

```
==> qemu: Provisioning with shell script: scripts/postfix.sh
    qemu: Reading package lists...
    qemu: Building dependency tree...
    qemu: Reading state information...
    qemu: The following additional packages will be installed:
    qemu:   guile-2.0-libs libgc1c2 libgsasl7 libkyotocabinet16v5 liblzo2-2
    qemu:   libmailutils5 libmysqlclient20 libntlm0 libpython2.7 libpython2.7-minimal
    qemu:   libpython2.7-stdlib mailutils-common mysql-common postfix ssl-cert
    qemu: Suggested packages:
    qemu:   mailutils-mh mailutils-doc procmail postfix-mysql postfix-pgsql postfix-ldap
    qemu:   postfix-pcre postfix-lmdb postfix-sqlite sasl2-bin dovecot-common resolvconf
    qemu:   postfix-cdb postfix-doc openssl-blacklist
    qemu: The following NEW packages will be installed:
    qemu:   guile-2.0-libs libgc1c2 libgsasl7 libkyotocabinet16v5 liblzo2-2
    qemu:   libmailutils5 libmysqlclient20 libntlm0 libpython2.7 libpython2.7-minimal
    qemu:   libpython2.7-stdlib mailutils mailutils-common mysql-common postfix ssl-cert
    qemu: 0 upgraded, 16 newly installed, 0 to remove and 0 not upgraded.
    qemu: Need to get 8,931 kB of archives.
```

- check_list.sh

==> qemu: Provisioning with shell script: **scripts/check_list.sh**

```
==> qemu: Provisioning with shell script: scripts/check_list.sh
    qemu: [PASS] Service is running : sshd
==> qemu: mailx: Null message body; hope that's ok
    qemu: [PASS] Service is running : Xvnc4
==> qemu: mailx: Null message body; hope that's ok
    qemu: [PASS] Port opend and listning: 22
==> qemu: mailx: Null message body; hope that's ok
    qemu: [PASS] Port opend and listning: 5901
==> qemu: mailx: Null message body; hope that's ok
    qemu: [PASS] User exist and has been added to sudoers: <sensitive>
==> qemu: mailx: Null message body; hope that's ok
    qemu: [PASS] User home directory exists: Path => /home/<sensitive>
==> qemu: mailx: Null message body; hope that's ok
    qemu: [SYSTEM] List of packages that has been added / updated
    qemu: 10,12c10,12
    qemu: < ii  apt                                  1.6.8                          amd64
    commandline package manager
    qemu: < ii  apt-utils                            1.6.8                          amd64
    package management related utility programs
    qemu: < ii  base-files                           10.1ubuntu2.4                  amd64
    Debian base system miscellaneous files
    qemu: ---
    qemu: > ii  apt                                  1.6.11                         amd64
    commandline package manager
    qemu: > ii  apt-utils                            1.6.11                         amd64
    package management related utility programs
    qemu: > ii  base-files                           10.1ubuntu2.5                  amd64
    Debian base system miscellaneous files
```

-> **Email notification** has been successfully **arrived**

☆	root 3	**[SYSTEM] List of packages that has been added / updated :: ubuntu03.saucelabs03.test** - 10,12c10,12 < ii apt 1.6.8 amd64 commandlin
☆	root 3	**[PASS] User home directory exists: Path => /home/vagrant :: ubuntu03.saucelabs03.test**
☆	root 3	**[PASS] User exist and has been added to sudoers: vagrant :: ubuntu03.saucelabs03.test**
☆	root 3	**[PASS] Port opend and listning: 5901 :: ubuntu03.saucelabs03.test**
☆	root 3	**[PASS] Port opend and listning: 22 :: ubuntu03.saucelabs03.test**
☆	root 3	**[PASS] Service is running : Xvnc4 :: ubuntu03.saucelabs03.test**
☆	root 3	**[PASS] Service is running : sshd :: ubuntu03.saucelabs03.test**

-> **Email notification** body

[SYSTEM] List of packages that has been added / updated :: ubuntu03.saucelabs03.test

Inbox ×

root <root@ubuntu03.saucelabs03.test>

to me

```
10,12c10,12
< ii  apt                    1.6.8                  amd64      commandline package manager
< ii  apt-utils              1.6.8                  amd64      package management related utility programs
< ii  base-files             10.1ubuntu2.4          amd64      Debian base system miscellaneous files
---
> ii  apt                    1.6.11                 amd64      commandline package manager
> ii  apt-utils              1.6.11                 amd64      package management related utility programs
> ii  base-files             10.1ubuntu2.5          amd64      Debian base system miscellaneous files
14c14
< ii  bash                   4.4.18-2ubuntu1        amd64      GNU Bourne Again SHell
---
> ii  bash                   4.4.18-2ubuntu1.2      amd64      GNU Bourne Again SHell
16c16
< ii  bind9-host             1:9.11.3+dfsg-1ubuntu1.3   amd64      DNS lookup utility (deprecated)
---
> ii  bind9-host             1:9.11.3+dfsg-1ubuntu1.8   amd64      DNS lookup utility (deprecated)
23,25c23,25
< ii  busybox-initramfs      1:1.27.2-2ubuntu3      amd64      Standalone shell setup for initramfs
< ii  busybox-static         1:1.27.2-2ubuntu3      amd64      Standalone rescue shell with tons of builtin utilities
< ii  bzip2                  1.0.6-8.1              amd64      high-quality block-sorting file compressor - utilities
```

- cleanup.sh

==> qemu: Provisioning with shell script: **scripts/cleanup.sh**

```
==> qemu: Provisioning with shell script: scripts/cleanup.sh
==> qemu:
==> qemu: WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
==> qemu:
    qemu: Reading package lists...
    qemu: Building dependency tree...
    qemu: Reading state information...
    qemu: 0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
==> qemu:
==> qemu: WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
==> qemu:
    qemu: Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease
    qemu: Hit:2 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease
    qemu: Hit:3 http://security.ubuntu.com/ubuntu bionic-security InRelease
    qemu: Hit:4 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease
    qemu: Reading package lists...
    qemu: Building dependency tree...
    qemu: Reading state information...
    qemu: All packages are up to date.
==> qemu: dd: error writing '/EMPTY': No space left on device
==> qemu: 3996+0 records in
==> qemu: 3995+0 records out
==> qemu: 4189224960 bytes (4.2 GB, 3.9 GiB) copied, 124.617 s, 33.6 MB/s
```

-> Output file

```
==> qemu: Gracefully halting virtual machine...
==> qemu: Converting hard drive...
Build 'qemu' finished.

==> Builds finished. The artifacts of successful builds are:
--> qemu: VM files in directory: output-virtualbox-qemu
[hjang@sw07 Linux]$ ^C
[hjang@sw07 Linux]$
[hjang@sw07 Linux]$
[hjang@sw07 Linux]$ ls output-virtualbox-qemu/
packer-ubuntu-18.04.2-amd64-qemu
[hjang@sw07 Linux]$
```

4. **Test - Manual**

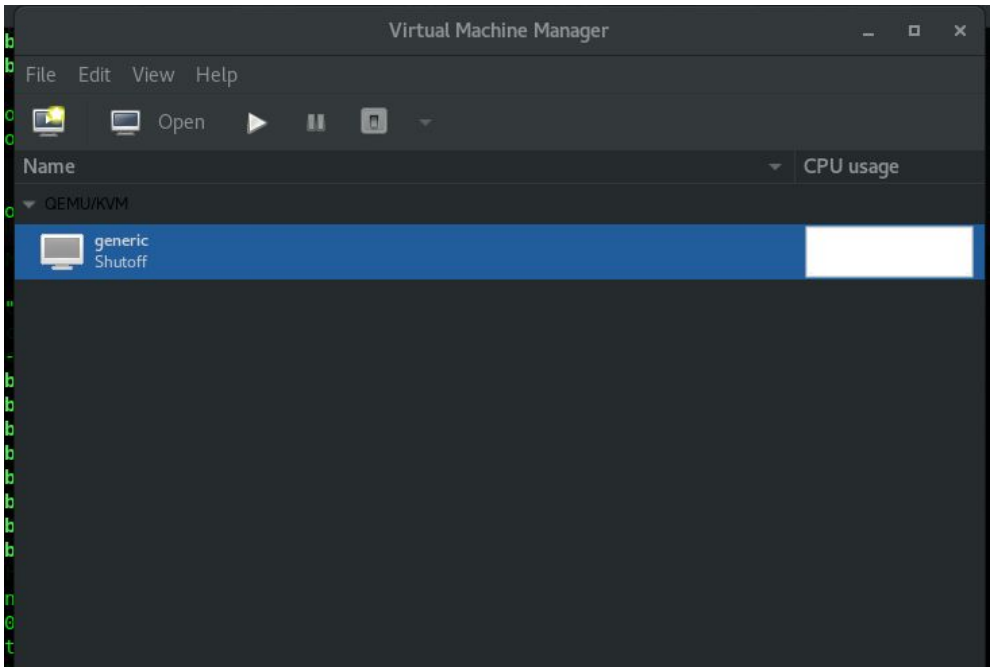
- Since **check_list.sh** assured that all automation part has been verified, we still need to check **VNC server** to connect VM from other hosts

- **Check Point**
  - Port 22 => **sshd**
  - Port 5901 => **vnc4server** is keep running even after reboot
    - Connectivity check from host machine
    - Firewall check
  - Custom user home directory : /home/vagrant
  - Custom user added in sudoer group : /etc/group
  - Timezone setting : UTC
  - System software up to date : apt update

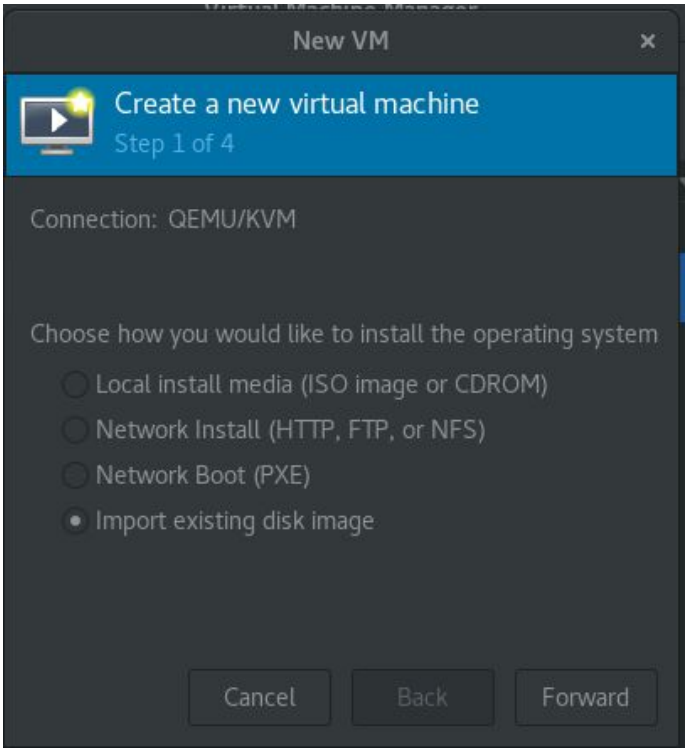
 # Run VM in **QEMU/KVM**

- Run **QEMU/KVM** on host machine

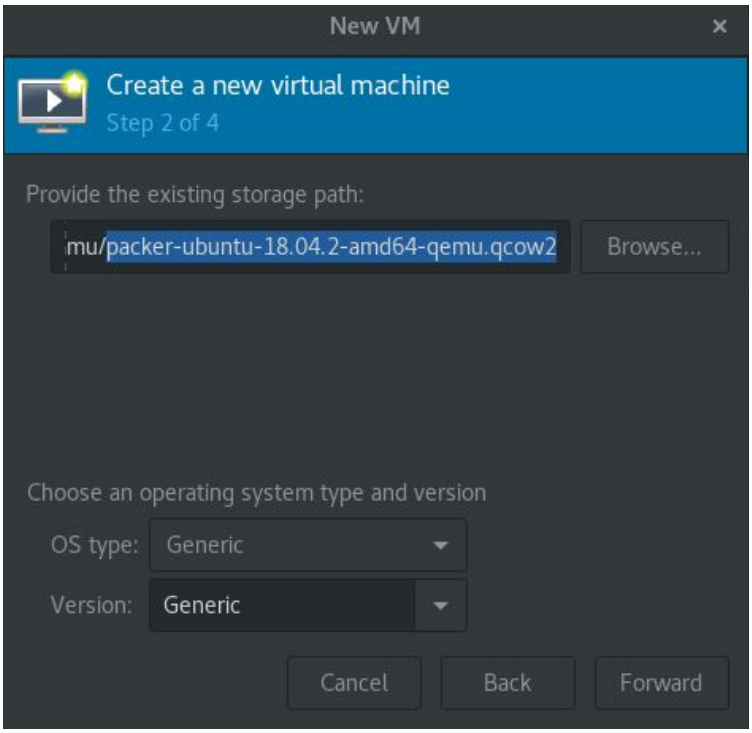
virt-manager



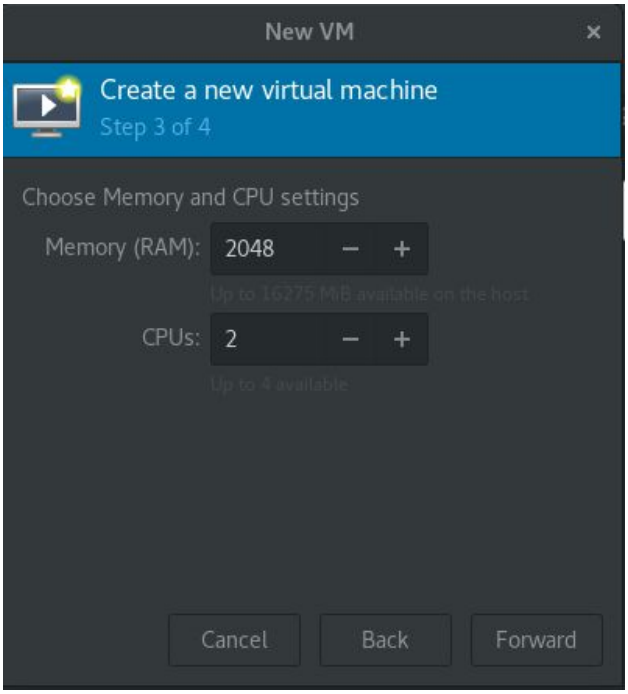
-> Import existing disk image



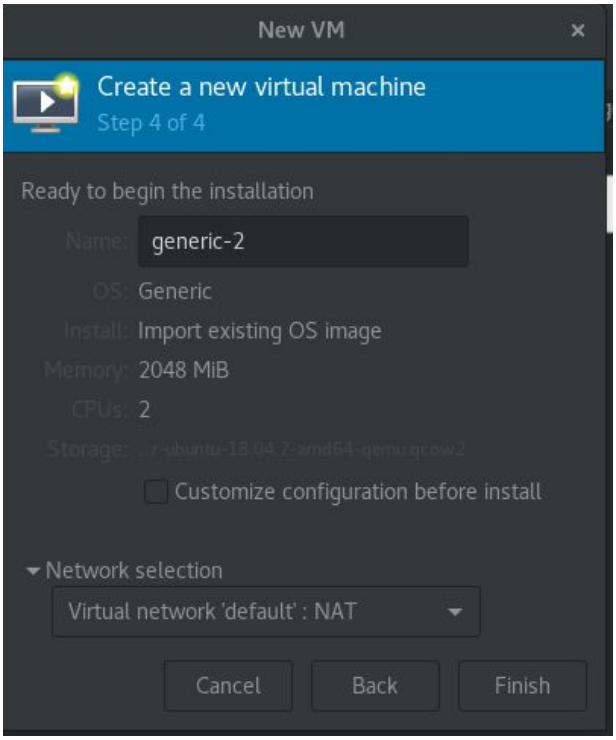
-> Provide the existing storage path



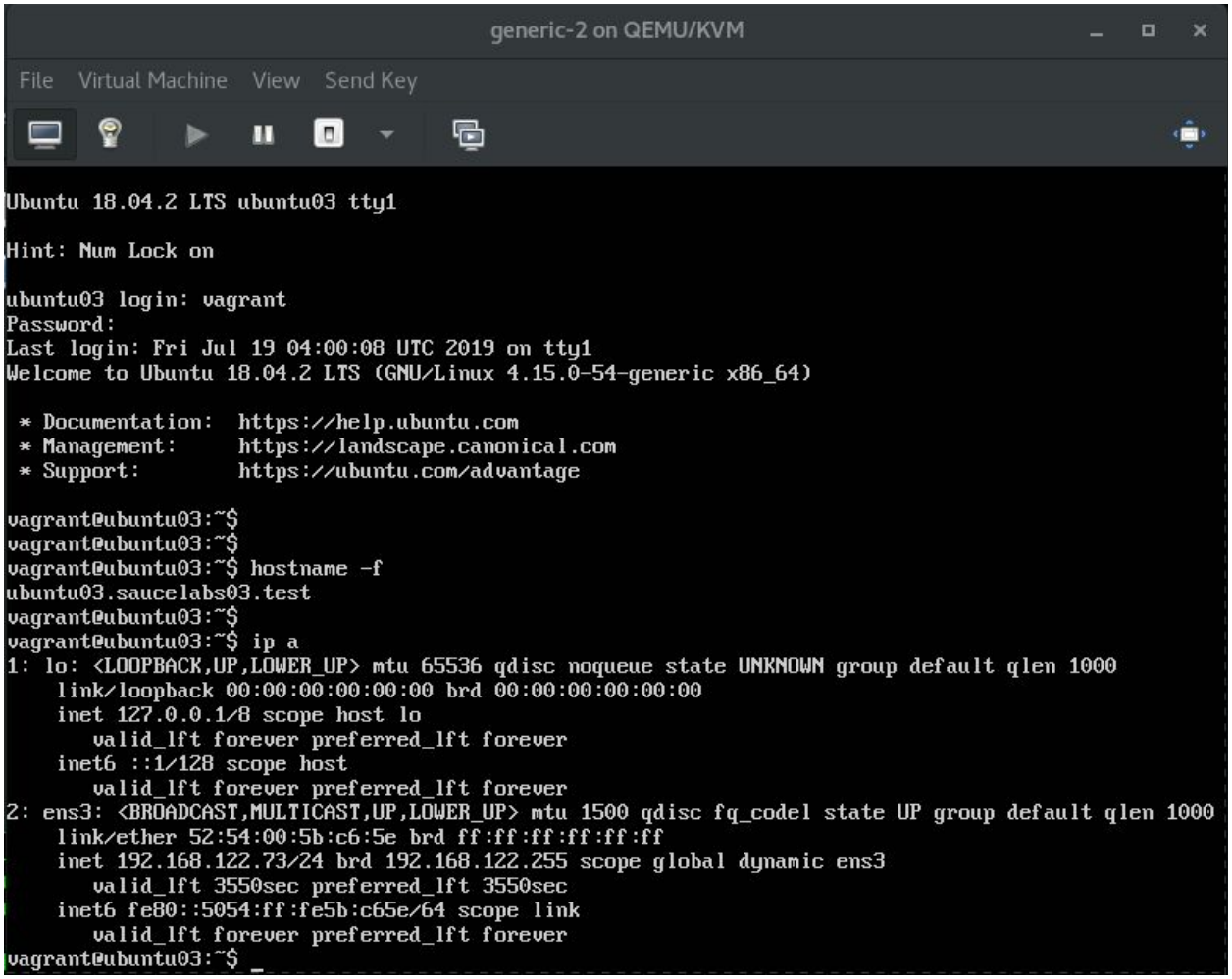
-> VM HW settings



-> Network setting **: NAT**



-> New VM has been launched with **QEMU/KVM**



- **Check Point**
  - Port 22 => **sshd**
  - Port 5901 => **vnc4server** is keep running even after reboot
    - Connectivity check from host machine
    - Firewall check

-> Port Check

ss -lnt

LISTEN	0	128	0.0.0.0**:22**	=> **sshd**
LISTEN	0	5	***:5901**	=> **vnc4server**

```
vagrant@ubuntu03:~$ ss -lnt
State       Recv-Q      Send-Q          Local Address:Port          Peer Address:Port
LISTEN      0           128                   0.0.0.0:111                 0.0.0.0:*
LISTEN      0           128                   0.0.0.0:6001                0.0.0.0:*
LISTEN      0           128             127.0.0.53%lo:53                  0.0.0.0:*
LISTEN      0           128                   0.0.0.0:22                  0.0.0.0:*
LISTEN      0           100                   0.0.0.0:25                  0.0.0.0:*
LISTEN      0           5                           *:5901                      *:*
LISTEN      0           128                      [::]:111                    [::]:*
LISTEN      0           100                      [::]:25                     [::]:*
vagrant@ubuntu03:~$
```

-> Firewall Check

```
vagrant@ubuntu03:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    Anywhere
5901/tcp                   ALLOW IN    Anywhere
22/tcp (v6)                ALLOW IN    Anywhere (v6)
5901/tcp (v6)              ALLOW IN    Anywhere (v6)

vagrant@ubuntu03:~$
```

-> Connectivity Check

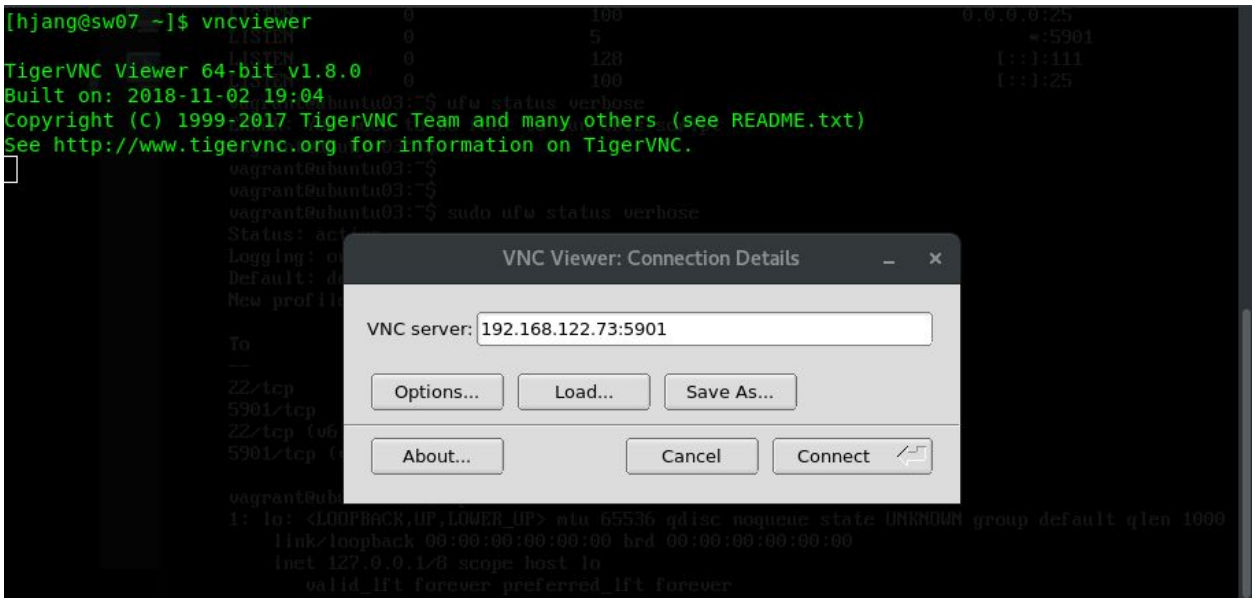
- sshd

```
[hjang@sw07 Linux]$ ssh vagrant@192.168.122.73
The authenticity of host '192.168.122.73 (192.168.122.73)' can't be established.
ECDSA key fingerprint is SHA256:4MA9G2jBrLzcLSAIovaKI0Xl8Q/rCelV9nCmFsLRnio.
ECDSA key fingerprint is MD5:6a:d1:63:97:30:ab:be:65:66:24:bc:1c:cc:de:3b:cc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.122.73' (ECDSA) to the list of known hosts.
vagrant@192.168.122.73's password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Last login: Fri Jul 19 09:14:11 2019
vagrant@ubuntu03:~$
vagrant@ubuntu03:~$
vagrant@ubuntu03:~$ hostname
ubuntu03
vagrant@ubuntu03:~$ hostname -f
ubuntu03.saucelabs03.test
vagrant@ubuntu03:~$
vagrant@ubuntu03:~$ whoami
vagrant
vagrant@ubuntu03:~$
vagrant@ubuntu03:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 52:54:00:5b:c6:5e brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.73/24 brd 192.168.122.255 scope global dynamic ens3
       valid_lft 3206sec preferred_lft 3206sec
    inet6 fe80::5054:ff:fe5b:c65e/64 scope link
       valid_lft forever preferred_lft forever
vagrant@ubuntu03:~$
```

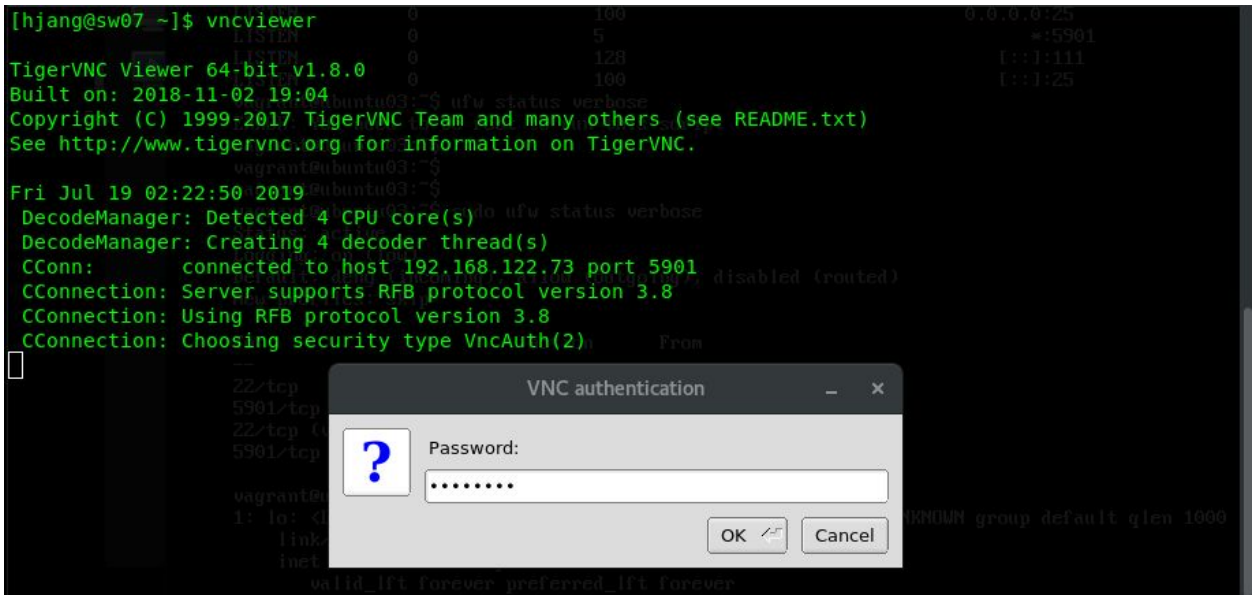
- vnc4server

vncviewer

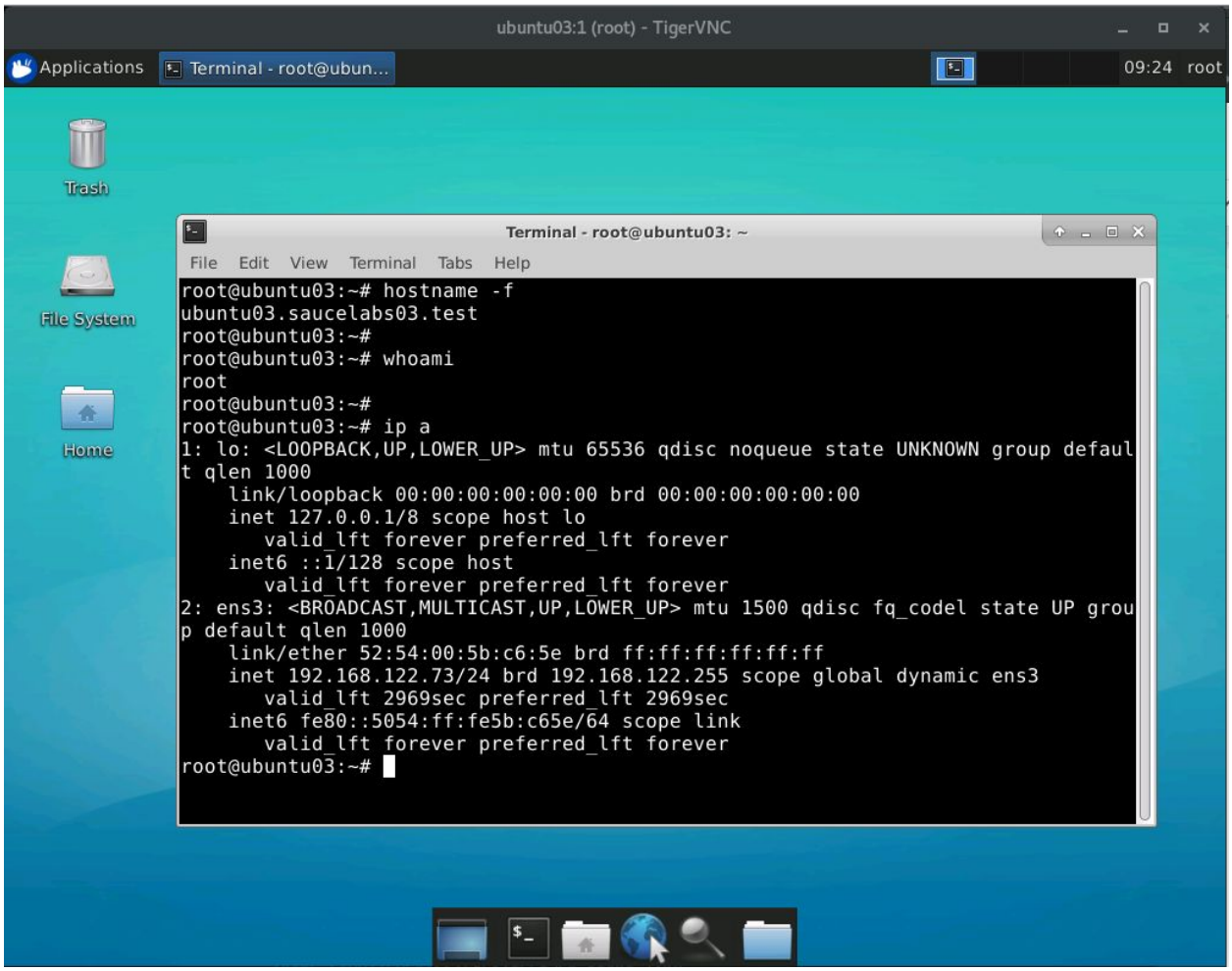
<IP_ADDR>:<VNC_PORT>



Password = **$vnc_password**
= P@ssw0rd



-> **VNC session** has been **established** as **root**



- Custom user home directory : **/home/vagrant**

```
vagrant@ubuntu03:~$ ll /home/
total 12
drwxr-xr-x  3 root    root    4096 Jul 19 03:43 ./
drwxr-xr-x 22 root    root    4096 Jul 19 03:56 ../
drwxr-xr-x  8 vagrant vagrant 4096 Jul 19 03:56 vagrant/
vagrant@ubuntu03:~$
vagrant@ubuntu03:~$ ll /home/vagrant/
total 52
drwxr-xr-x 8 vagrant vagrant 4096 Jul 19 03:56 ./
drwxr-xr-x 3 root    root    4096 Jul 19 03:43 ../
-rw------- 1 vagrant vagrant    8 Jul 19 03:56 .bash_history
-rw-r--r-- 1 vagrant vagrant  220 Jul 19 03:43 .bash_logout
-rw-r--r-- 1 vagrant vagrant 3771 Jul 19 03:43 .bashrc
drwx------ 4 vagrant vagrant 4096 Jul 19 03:53 .cache/
drwx------ 3 root    root    4096 Jul 19 03:53 .config/
drwx------ 3 root    root    4096 Jul 19 03:53 .dbus/
drwx------ 3 vagrant vagrant 4096 Jul 19 03:46 .gnupg/
drwxr-xr-x 3 root    root    4096 Jul 19 03:53 .local/
-rw-r--r-- 1 vagrant vagrant  807 Jul 19 03:43 .profile
-rw-r--r-- 1 vagrant vagrant    0 Jul 19 03:46 .sudo_as_admin_successful
drwxr-xr-x 2 root    root    4096 Jul 19 03:53 .vnc/
-rw------- 1 root    root     102 Jul 19 03:53 .Xauthority
vagrant@ubuntu03:~$
```

- Custom user added in sudoer group : **/etc/group**

```
vagrant@ubuntu03:~$ cat /etc/group | grep vagrant
sudo:x:27:vagrant
vagrant:x:900:
vagrant@ubuntu03:~$
```

- Timezone setting : **UTC**

```
vagrant@ubuntu03:~$ timedatectl
                      Local time: Fri 2019-07-19 09:27:13 UTC
                  Universal time: Fri 2019-07-19 09:27:13 UTC
                        RTC time: Fri 2019-07-19 09:27:14
                       Time zone: Etc/UTC (UTC, +0000)
       System clock synchronized: yes
systemd-timesyncd.service active: yes
                 RTC in local TZ: no
vagrant@ubuntu03:~$
```

- System software up to date : apt update

```
vagrant@ubuntu03:~$ sudo apt update
Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [684 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu bionic-updates/main i386 Packages [560 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [974 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe i386 Packages [958 kB]
Fetched 3,428 kB in 1s (2,447 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
vagrant@ubuntu03:~$
```

5. **Test - Vagrant**

- Automate **importing** and **running** VM procedure with **Vagrant**
- **Vagrant** can do more than just import/run VMs!

-> In this section, we'll use template as below to demonstrate bring up VM in **Oracle VirtualBox** with **Vagrant**

ubuntu_18.04.2_**vbox_w_vagrant**.json

```
"variables": {
  "get_domain": "saucelabs02.test",
  "get_hostname": "ubuntu02",
  "ssh_username": "vagrant",
  "ssh_password": "vagrant",
  "sudoer_name": "vagrant",
  "http_directory": "http",
  "output_directory": "output-virtualbox-vbox_vg",
  "vm_name": "packer-ubuntu-18.04.2-amd64-vbox_vg",
  "guest_os_type": "Ubuntu_64",
  "disk_size": "8192",
  "vboxmanage_memory": "4092",
  "vboxmanage_cpu": "2",
  "vlan_name": "enp0s31f6.2000",
  "output_vg_box": "ubuntu-18.04.2-vbox.box",

  "iso_checksum_type": "file",
  "iso_checksum_url": "http://cdimage.ubuntu.com/ubuntu/releases/bionic/release/SHA256SUMS",
  "iso_urls": "http://cdimage.ubuntu.com/ubuntu/releases/bionic/release/ubuntu-18.04.2-server-amd64.iso"
},
  "sensitive-variables": ["ssh_username", "ssh_password"],
```

-> Pay attention on “**builders**” block to see what’s different between **QEMU/KVM** template and **VirtualBox** template

"type": "**virtualbox-iso**",

```
26
27      "builders": [
28          {
29              "type": "virtualbox-iso",
30              "boot_command": [
31                  "<esc><wait>",
32                  "<esc><wait>",
33                  "<enter><wait>",
34                  "/install/vmlinuz<wait>",
35                  " auto<wait>",
36                  " console-setup/ask_detect=false<wait>",
```

" grub-installer/bootdev=/dev/**sda**<wait>",

```
"builders": [
    {
        "type": "virtualbox-iso",
        "boot_command": [
            "<esc><wait>",
            "<esc><wait>",
            "<enter><wait>",
            "/install/vmlinuz<wait>",
            " auto<wait>",
            " console-setup/ask_detect=false<wait>",
            " console-setup/layoutcode=us<wait>",
            " console-setup/modelcode=pc105<wait>",
            " debconf/frontend=noninteractive<wait>",
            " debian-installer=en_US<wait>",
            " fb=false<wait>",
            " initrd=/install/initrd.gz<wait>",
            " kbd-chooser/method=us<wait>",
            " keyboard-configuration/layout=USA<wait>",
            " keyboard-configuration/variant=USA<wait>",
            " locale=en_US<wait>",
            " netcfg/get_domain={{user `get_domain`}}<wait>",
            " netcfg/get_hostname={{user `get_hostname`}}<wait>",
            " grub-installer/bootdev=/dev/sda<wait>",
            " noapic<wait>",
            " preseed/url=http://{{ .HTTPIP }}:{{ .HTTPPort }}/preseed.cfg<wait>",
            " -- <wait>",
            "<enter><wait>"
        ],
```

"**vboxmanage**": [
["**modifyvm**", "{{.Name}}", "**--memory**", "{{user `vboxmanage_memory`}}"],
["**modifyvm**", "{{.Name}}", "**--cpus**", "{{user `vboxmanage_cpu`}}"]
]

```
            "iso_checksum_type": "{{user `iso_checksum_type`}}",
            "iso_checksum_url": "{{user `iso_checksum_url`}}",
            "guest_additions_path": "VBoxGuestAdditions_{{.Version}}.iso",
            "shutdown_command": "echo '{{user `sudoer_name`}}'|sudo -S shutdown -P now",
            "virtualbox_version_file": ".vbox_version",
            "vm_name": "{{user `vm_name`}}",
            "vboxmanage": [
            ["modifyvm", "{{.Name}}", "--memory", "{{user `vboxmanage_memory`}}"],
            ["modifyvm", "{{.Name}}", "--cpus", "{{user `vboxmanage_cpu`}}"]
            ]
        }
    ],
```

-> Run **Packer** to build image

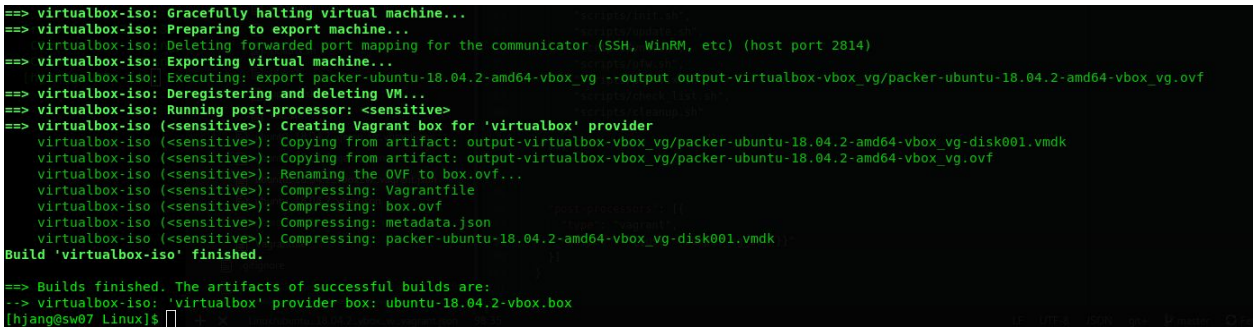
packer build <FILE_PATH>

```
[hjang@sw07 Linux]$ packer build ubuntu_18.04.2_vbox_w_vagrant.json
virtualbox-iso output will be in this color.

==> virtualbox-iso: Retrieving Guest additions
==> virtualbox-iso: Trying /usr/share/virtualbox/VBoxGuestAdditions.iso
==> virtualbox-iso: Trying /usr/share/virtualbox/VBoxGuestAdditions.iso
==> virtualbox-iso: /usr/share/virtualbox/VBoxGuestAdditions.iso => /home/hjang/SauceLabs_CC/Linux/packer_cache/26fb5ce5
==> virtualbox-iso: leaving retrieve loop for Guest additions
==> virtualbox-iso: Retrieving ISO
==> virtualbox-iso: Trying http://cdimage.ubuntu.com/ubuntu/releases/bionic/release/ubuntu-18.04.2-server-amd64.iso
==> virtualbox-iso: Trying http://cdimage.ubuntu.com/ubuntu/releases/bionic/release/ubuntu-18.04.2-server-amd64.iso?check
om%2Fubuntu%2Freleases%2Fbionic%2Frelease%2FSHA256SUMS
SHA256SUMS 757 B / 757 B [==========================================================================================
```

-> Vagrant box file has been created

- If you run **regular** template without “post-processor” block, you’ll get .**ovf & .vmdk** for **VirtualBox** output instead of **.box** for **Vagrant**

==> virtualbox-iso: Exporting virtual machine...
virtualbox-iso: Executing: export packer-ubuntu-18.04.2-amd64-vbox_vg --output output-virtualbox-vbox_vg/**packer-ubuntu-18.04.2-amd64-vbox_vg.ovf**
==> virtualbox-iso: Deregistering and deleting VM...
==> virtualbox-iso: Running post-processor: <sensitive>
==> virtualbox-iso (<sensitive>): Creating Vagrant box for 'virtualbox' provider
virtualbox-iso (<sensitive>): Copying from artifact: output-virtualbox-vbox_vg/**packer-ubuntu-18.04.2-amd64-vbox_vg-disk001.vmdk**
virtualbox-iso (<sensitive>): Copying from artifact: output-virtualbox-vbox_vg/packer-ubuntu-18.04.2-amd64-vbox_vg.ovf
virtualbox-iso (<sensitive>): Renaming the OVF to box.ovf...
virtualbox-iso (<sensitive>): Compressing: **Vagrantfile**
virtualbox-iso (<sensitive>): Compressing: **box.ovf**
virtualbox-iso (<sensitive>): Compressing: **metadata.json**
virtualbox-iso (<sensitive>): Compressing: packer-ubuntu-18.04.2-amd64-vbox_vg-disk001.vmdk
Build 'virtualbox-iso' finished.



-> Add **Vagrant box** in list

vagrant box add --name <u>default</u> ubuntu-18.04.2-vbox.box

```
[hjang@sw07 Linux]$ ls | grep *.box
ubuntu-18.04.2-vbox.box
[hjang@sw07 Linux]$
[hjang@sw07 Linux]$ vagrant box list
packer1 (aws, 0)
[hjang@sw07 Linux]$
[hjang@sw07 Linux]$
[hjang@sw07 Linux]$ vagrant box add --name default ubuntu-18.04.2-vbox.box
==> box: Box file was not detected as metadata. Adding it directly...
==> box: Adding box 'default' (v0) for provider:
    box: Unpacking necessary files from: file:///home/hjang/SauceLabs_CC/Linux/ubuntu-18.04.2-vbox.box
==> box: Successfully added box 'default' (v0) for 'virtualbox'!
[hjang@sw07 Linux]$
```

-> Try to up VM

vagrant up --provider=virtualbox

```
[hjang@sw07 Linux]$ vagrant up --provider=virtualbox
A Vagrant environment or target machine is required to run this
command. Run `vagrant init` to create a new Vagrant environment. Or,
get an ID of a target machine from `vagrant global-status` to run
this command on. A final option is to change to a directory with a
Vagrantfile and to try again.
[hjang@sw07 Linux]$
```

-> Need to run **vagrant init** to create **Vagrant environment**

- Vagrant will create **Vagrantfile** for you so you can modify it

vagrant init

```
[hjang@sw07 Linux]$ vagrant init
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
[hjang@sw07 Linux]$
[hjang@sw07 Linux]$ cat Vagrantfile
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure("2") do |config|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You can search for
  # boxes at https://vagrantcloud.com/search.
  config.vm.box = "base"
```

-> Modify **Vagrantfile**

-> **vbox name** has to be matched **<u>with added box</u>**

config.vm.box = "default"

```ruby
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure("2") do |config|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You can search for
  # boxes at https://vagrantcloud.com/search.
  config.vm.box = "default"

  # Disable automatic box update checking. If you disable this, then
  # boxes will only be checked for updates when the user runs
  # `vagrant box outdated`. This is not recommended.
  # config.vm.box_check_update = false

```

-> Enable network on your choice

config.vm.network "public_network"

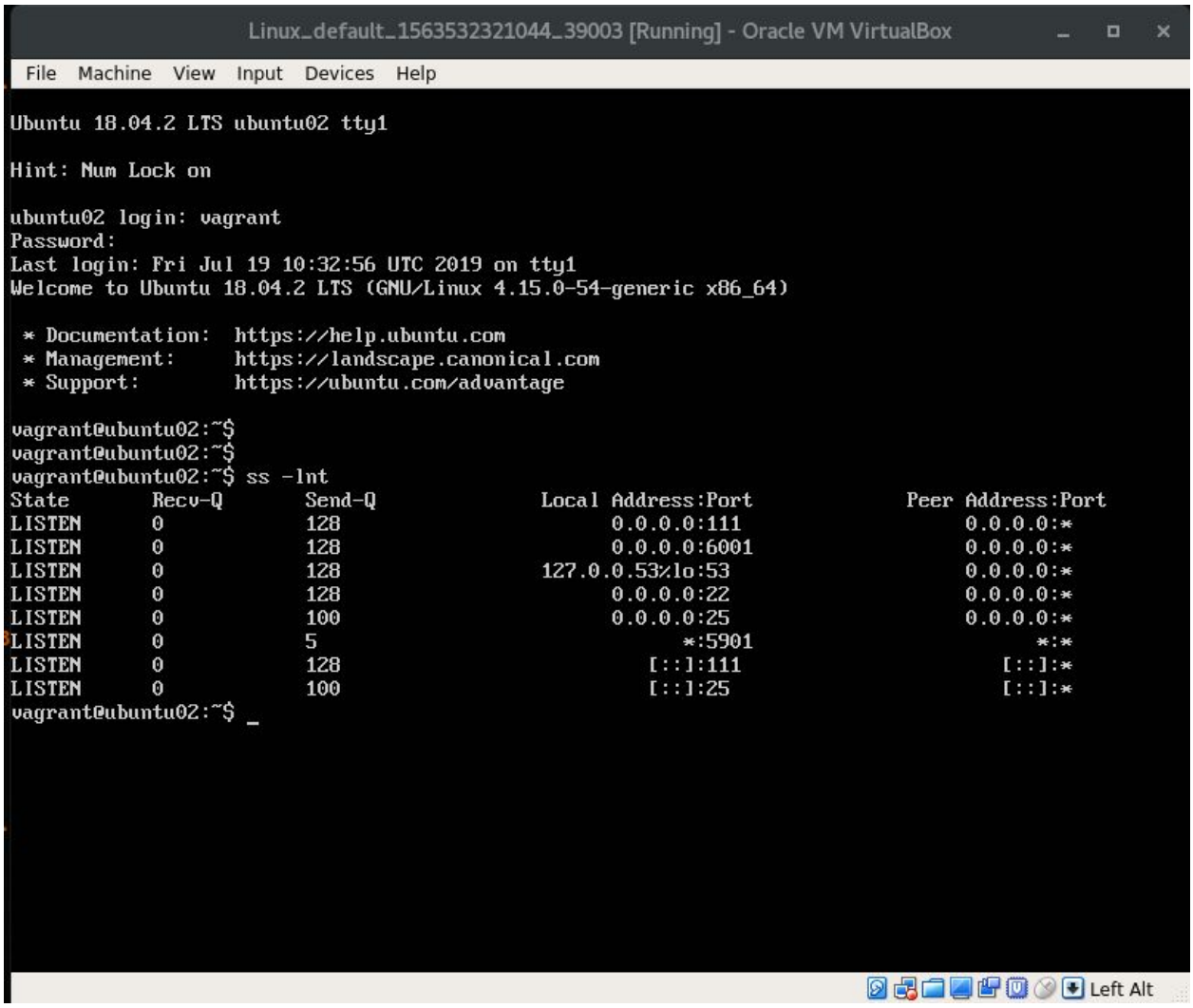
```
# Create a forwarded port mapping which allows access to a specific port
# within the machine from a port on the host machine. In the example below,
# accessing "localhost:8080" will access port 80 on the guest machine.
# NOTE: This will enable public access to the opened port
# config.vm.network "forwarded_port", guest: 80, host: 8080

# Create a forwarded port mapping which allows access to a specific port
# within the machine from a port on the host machine and only allow access
# via 127.0.0.1 to disable public access
# config.vm.network "forwarded_port", guest: 80, host: 8080, host_ip: "127.0.0.1"

# Create a private network, which allows host-only access to the machine
# using a specific IP.
# config.vm.network "private_network", ip: "192.168.33.10"

# Create a public network, which generally matched to bridged network.
# Bridged networks make the machine appear as another physical device on
# your network.
 config.vm.network "public_network"
```

-> Run **vagrant up** again

vagrant up --provider=virtualbox

```
[hjang@sw07 Linux]$ vagrant up --provider=virtualbox
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'default'...
==> default: Matching MAC address for NAT networking...
==> default: Setting the name of the VM: Linux_default_1563531999016_80102
==> default: Clearing any previously set network interfaces...
==> default: Available bridged network interfaces:
1) enp0s31f6
2) enp0s31f6.2000
3) vlan2110
4) vmnet8
5) vlan2114
6) vlan2115
7) virbr0
8) vlan2015
9) vlan2016
10) vlan2108
11) vlan2008
12) vmnet1
13) docker0
==> default: When choosing an interface, it is usually the one that is
==> default: being used to connect to the internet.
    default: Which interface should the network bridge to? 2
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
    default: Adapter 2: bridged
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
```

-> VM has been up in **VirtualBox**



-> To **delete** VM and remove from **Vagrant**

vagrant destroy

```
[hjang@sw07 Linux]$ vagrant destroy
    default: Are you sure you want to destroy the 'default' VM? [y/N] y
==> default: Forcing shutdown of VM...
==> default: Destroying VM and associated drives...
[hjang@sw07 Linux]$
```

[Reference]

Vagrant > Basic Provider Usage
https://www.vagrantup.com/docs/providers/basic_usage.html

Packer > Vagrant Boxes
https://www.packer.io/intro/getting-started/vagrant.html

How to use Packer to create Ubuntu 18.04 Vagrant boxes
https://www.serverlab.ca/tutorials/dev-ops/automation/how-to-use-packer-to-create-ubuntu-18-04-vagrant-boxes/

 # Test - Inspec

- Create a template in **Ruby** to run test with **Inspec** against target host while its running

```
[hjang@sw07 Linux]$ tree check_list/
check_list/
|-- controls
|   `-- check_list.rb
|-- inspec.lock
|-- inspec.yml
`-- README.md

1 directory, 4 files
[hjang@sw07 Linux]$
```

- controls/**check_list.rb**
    - Actual **audit rules** are defined in this file

```ruby
##############################################
# 0. Is package installed? : vnc4server
describe package('vnc4server'), :if => os[:family] == 'ubuntu' do
  it { should be_installed }
end


##############################################
# 1. Is service running? : sshd
# 2. Is service running? : vnc4server
describe service('ufw') do
  it { should be_enabled }
  it { should be_running }
end

describe service('sshd') do
  it { should be_enabled }
  it { should be_running }
end
describe processes("Xvnc4") do
  its('users') { should eq ['root'] }
end
```

```ruby
##########################################
# 3. Is port opend? : sshd
# 4. Is port opend? : vnc4server
describe port(22) do
  it { should be_listening }
end
describe port(5901) do
  it { should be_listening }
end


##########################################
# 5. New user added and sudoers?
# 6. Does user has home directory?
describe user('vagrant') do
  it { should exist }
  its('uid') { should eq 900 }
  its('groups') { should eq ['vagrant', 'sudo']}
  its('home') { should eq '/home/vagrant' }
  its('shell') { should eq '/bin/bash' }
end
```

```ruby
##########################################
# 7. Is timezone set by UTC?
describe command('cat /etc/timezone && timedatectl | grep "Time zone" | cut -d " " -f 19') do
  its('stdout') { should match 'Etc/UTC' }
end


##########################################
# 8. Is vnc emulator has correct symlink?
describe file('/etc/alternatives/x-terminal-emulator') do
    it { should exist }
    it { should be_symlink }
    it { should be_file }
    it { should be_linked_to '/usr/bin/xfce4-terminal.wrapper' }
end



##########################################
# 9. Is cron_vnc.sh in /etc/cron.d/cron_vnc?
describe file('/etc/cron.d/cron_vnc') do
    it { should exist }
    its('content') { should match( '@reboot root /root/crontabs/cron_vnc.sh' ) }
end
```

\

- **Inspec.yml**
    - **Metadata** of controls

```yaml
name: check_list
title: Check list for target host
summary: An InSpec Profile to check server status
version: 0.1.0
```

-> Run test with **Inspec**

inspec exec check_list **-t ssh:**//<USER_NAME>**:**<PASS_WORD>**@**<TARGET_IP>

```
[hjang@sw07 Linux]$ inspec exec check_list -t ssh://vagrant:vagrant@192.168.122.29
[2019-07-19T07:23:16-07:00] WARN: DEPRECATION: The service `be_running?` matcher is deprecated. This is only allowed for compatibility wi
xample.rb:13)
[2019-07-19T07:23:17-07:00] WARN: DEPRECATION: The service `be_running?` matcher is deprecated. This is only allowed for compatibility wi
xample.rb:18)

Profile: Check list for target host (check_list)
Version: 0.1.0
Target:  ssh://vagrant@192.168.122.29:22

  System Package vnc4server
     ✔  should be installed
  Service ufw
     ✔  should be enabled
     ✔  should be running
  Service sshd
     ✔  should be enabled
     ✔  should be running
  Processes Xvnc4
     ✔  users should eq ["root"]
  Port 22
     ✔  should be listening
  Port 5901
     ✔  should be listening
  User vagrant
     ✔  should exist
     ✔  uid should eq 900
     ✔  groups should eq ["vagrant", "sudo"]
     ✔  home should eq "/home/vagrant"
     ✔  shell should eq "/bin/bash"
  Command: `cat /etc/timezone && timedatectl | grep "Time zone" | cut -d " " -f 19`
     ✔  stdout should match "Etc/UTC"
  File /etc/alternatives/x-terminal-emulator
     ✔  should exist
     ✔  should be symlink
     ✔  should be file
     ✔  should be linked to "/usr/bin/xfce4-terminal.wrapper"
  File /etc/cron.d/cron_vnc
     ✔  should exist
     ✔  content should match "@reboot root /root/crontabs/cron_vnc.sh"

Test Summary: 20 successful, 0 failures, 0 skipped
[hjang@sw07 Linux]$
```

[Reference]

Learn Chef > Try InSpec
https://learn.chef.io/modules/try-inspec#/

Inspec > command
https://www.inspec.io/docs/reference/resources/command/

Inspec > file
https://www.inspec.io/docs/reference/resources/file/

6. Questions

- **How can I reduce building time when provisioning OS image?**
  - Can I use **<u>tags</u>** in Packer? (Like **Ansible**)

    -> **[Assumption]**

    Use Ansible to provisioning last image ->

    Modify code -> Build image again? **=> Efficiency?**

- **How to build multiple images at the same time?**
  - Packer > Parallel Builds
    https://www.packer.io/intro/getting-started/parallel-builds.html

- **What's the best combination to build container image?**
  - Packer + Ansible - Dockerfile = AwesomeContainer
    https://alex.dzyoba.com/blog/packer-for-docker/

7. **Resources**

What is KVM?
https://www.redhat.com/en/topics/virtualization/what-is-KVM

Install KVM Hypervisor on CentOS 7.x and RHEL 7.x
https://www.linuxtechi.com/install-kvm-hypervisor-on-centos-7-and-rhel-7/

How to create a CentOS 7 KVM image with Packer
https://velenux.wordpress.com/2016/11/13/how-to-create-a-centos-7-kvm-image-with-packer/

qemu-system-x86_64
https://askubuntu.com/questions/138140/how-do-i-install-qemu

QEMU Builder
https://packer.io/docs/builders/qemu.html#qemu_binary

QEMU > -m [size=]megs[,slots=n,maxmem=size]
https://qemu.weilnetz.de/doc/qemu-doc.html

packer-templates/centos-7/packer.json
https://github.com/lightcode/packer-templates/blob/master/centos-7/packer.json

SauceLabs Coding Challenge

[Starts at 20:30PM PDT, 170719 ~ Ends at 20:30PM PDT, 190719]

- Journal -