**CLOUD SOFTWARE**

**Google Compute Engine**

Founded in 1998 in USA.

**Google Compute Engine** is a service that provides virtual machines that run on **Google** infrastructure.

It offers great networking infrastructure, performing fast and saving cost.

This is very flexible with operating systems such as Windows and Linux; and open platforms such as Word press, PHP and Dot Net.

Google Compute Engine lets you run large-scale computing workloads on the same infrastructure that runs Google Search, Gmail and Ads.

We can launch virtual machines on-demand; manage network connectivity using a simple but from your virtual machines.

Google Compute Engine is in limited preview; please visit cloud.google.com to learn how to sign up.

**Google Compute Engine provides the following features:**

Full Virtual Machines, Kernel-hosted virtual machines running on Launch 1, 2, 4, or 8 virtual core instances with 3.75GB of memory per virtual core.

We can also create snapshots of our disks for backup/restore purposes, and can mount these devices in a mode that allows multiple virtual machines to read from a single device.

Flexible networking you to connect your virtual machines to each other and to the Internet.

* Isolation Strong controls in our network stack protect our customers
* External IP addresses Connect your virtual machine to the Internet with either static IP addresses or ephemeral addresses that are assigned to your machines.
* Control who can talk to which virtual machine(s)

## Google Compute Engine Offers

When a new player like Google comes along, those who have been looking for a feature that they could not get from their current providers will want to check out the new kid on the block. Google of course is a well known brand and has deep pockets to compete with the best in the cloud.

 However, offering new features like Load-balancing will not make customers come running in droves.  Pricing is also not enough to entice people to make a shift from their current provider to Google Compute Engine (GCE).

If you want to build your own collection of Linux boxes, Google Compute Engine offers a nice, generic way to buy servers at what -- depending on the size of compute instance you need -- can be a great price.

**Installation Steps:**

To build the image, I’ve used the following:

* Ubuntu 14.04 Desktop (will host our virtual machine then-to-become GCE image)
* Qemu emulator/virtualizer
* Google Cloud SDK

We will begin with the infrastructure set-up. Start the Ubuntu Desktop (of course you can use any other dist).

**Step 1 - Install Qemu and Ubuntu Virtual Machine**

There are several options to get it done, I’ll just pull it from the repository:

$ sudo apt-get install qemu -y

For the sake of order, create a folder which will contain the virtual machine and other stuff:

$ mkdir VM

$ mkdir VM/Disks/SourceISO

$ mkdir VM/BackupImage

Get your Ubuntu 14.04 LTS ISO from here, and download it to the SourceISO folder

$ wget -P VM/SourceISO http://releases.ubuntu.com/14.04/ubuntu-14.04-server-amd64.iso

Create a disk for the virtual machine:

$ qemu-img create VM/Disks/ubuntu.raw 10g

Let’s start the virtual machine and boot it from the ISO:

$ qemu-system-x86\_64 --enable-kvm -smp 1 -m 512m -net nic,model=virtio -net user,hostfwd=tcp::2222-:22 -device virtio-scsi-pci,id=scsi -device scsi-hd,drive=hd,physical\_block\_size=4096 -drive if=none,id=hd,file=VM/Disks/ubuntu.raw,cache=none -cdrom VM/SourceISO/ubuntu-14.04-server-amd64.iso

This will invoke an x64 instance with the required hardware and boot into the setup. Go ahead and install the machine, customize it any way you want, as long as you do these:

* There should be only one partition (manual setup)
* Don’t encrypt home dir
* Select UTC time zone
* Install OpenSSH server

Your qemu instance of Ubuntu will finish installation and reboot. You’re half-way there! :)

**Step 2 - Update the Image & Install Packaging tools**

Now, let’s go ahead and set-up the Ubuntu image itself. You can work on the qemu console window if you wish, the drawback is the lack of copy/paste, therefore I would recommend the SSH session.

Connect to the instance with the user you’d created during the installation:

$ ssh -p 2222 user@127.0.0.1

Let’s make sure the virtual machine is up to date:

$ sudo su -l

$ apt-get update

$ apt-get upgrade -u -y

At this point, I would recommend to take a backup copy of your image (it will also be useful for future images creation, you’ll have a clean up to date installation to manipulate at the future).

Shutdown the machine and just make a copy of the ubuntu.raw file to a different location, for example:

$ cp VM/Disks/ubuntu.raw mkdir VM/BackupImage/ubuntu\_clean.raw

Once done, turn-on the instance again.

To package the virtual machine as GCE’s image, we will need to install the latest Google packages from git (you’ll need to have root privileges for this)

$ apt-get install kpartx ethtool

$ wget https://github.com/GoogleCloudPlatform/compute-image-packages/releases/download/1.1.2/python-gcimagebundle\_1.1.2-1\_all.deb https://github.com/GoogleCloudPlatform/compute-image-packages/releases/download/1.1.2/google-compute-daemon\_1.1.2-1\_all.deb https://github.com/GoogleCloudPlatform/compute-image-packages/releases/download/1.1.2/google-startup-scripts\_1.1.2-1\_all.deb

$ dpkg -i google-compute-daemon\_1.1.2-1\_all.deb google-startup-scripts\_1.1.2-1\_all.deb python-gcimagebundle\_1.1.2-1\_all.deb

**Step 3 - Image Configuration & Upload to GCE**

If you didn’t set the instance as UTC at setup, do it now:

$ ln -sf /usr/share/zoneinfo/UTC /etc/localtime

$ apt-get install ntp -y

You’ll need to comment-out Ubuntu NTP server and add GCEs - server metadata.google.internal

$ sed -i 's/^server ntp.ubuntu.com/# server ntp.ubuntu.com\nserver metadata.google.internal/' /etc/ntp.conf

Then, remove the /etc/hosts file and add the Google’s metadata server IP address

$ rm /etc/hostname

$ echo "169.254.169.254 metadata.google.internal metadata" >> /etc/hosts

$ ln -s /usr/share/google/set-hostname /etc/dhcp/dhclient-exit-hooks.d/

Create /etc/init/ttyS0 file with the following content:

$ vim /etc/init/ttyS0.conf

# ttyS0 - getty

start on stopped rc or RUNLEVEL=[2345]

stop on runlevel [!2345]

respawn

exec /sbin/getty -L 115200 ttyS0 vt102

Edit the /etc/default/grub file to include these settings:

GRUB\_CMDLINE\_LINUX="console=ttyS0,115200n8 ignore\_loglevel"

GRUB\_SERIAL\_COMMAND="serial --speed=115200 --unit=0 --word=8 --parity=no --stop=1"

GRUB\_TERMINAL=console

$ sed -i 's/^GRUB\_CMDLINE\_LINUX=""/GRUB\_CMDLINE\_LINUX="console=ttyS0,115200n8 ignore\_loglevel"\nGRUB\_SERIAL\_COMMAND="serial --speed=115200 --unit=0 --word=8 --parity=no --stop=1"/' /etc/default/grub

$ sed -i 's/^#GRUB\_TERMINAL=console/GRUB\_TERMINAL=console/' /etc/default/grub

And update grub:

$ update-grub2

For FIPS mode (optional) execute:

$ mkdir /etc/gcrypt && echo "1" >> /etc/gcrypt/fips\_enabled

As with all security and installation preferences, it comes down to what you need and how you like to get it. I would recommend going over Google’s [recommendations](https://developers.google.com/compute/docs/images#ssh) and see what fits your need. In this manual,  I’ve kept it very basic.

Recommended that those will be Disabled - root ssh login, password authentication, host based authentication - at /etc/ssh/sshd\_config

Add GOOGLE to the sshd\_not\_to\_be\_run

$ echo "GOOGLE" > /etc/ssh/sshd\_not\_to\_be\_run

Remove old host SSH keys.

**Important** - by doing this you won't be able to SSH to the machine, only console access until you’ll upload the image to GCE, so please be careful :)

$ rm /etc/ssh/ssh\_host\_\*

Create /etc/sysctl.conf.d/12-gce-recommended.conf with the following content:

$ vim/etc/sysctl.d/12-gce-recommended.conf

# provides protection from ToCToU races

fs.protected\_hardlinks=1

# provides protection from ToCToU races

fs.protected\_symlinks=1

# makes locating kernel addresses more difficult

kernel.kptr\_restrict=1

# set ptrace protections

kernel.yama.ptrace\_scope=1

# set perf only available to root

kernel.perf\_event\_paranoid=2

# disable ipv6

net.ipv6.conf.all.disable\_ipv6=1

net.ipv6.conf.default.disable\_ipv6=1

net.ipv6.conf.lo.disable\_ipv6=1

Great! Now, we just need to prepare the bundle of image and upload it to GCE. First check if /usr/lib/python2.7/dist-packages/gcimagebundlelib/manifest.py require patching by [this](https://github.com/GoogleCloudPlatform/compute-image-packages/pull/32/files) link.

You'll need gsutil and gcutil in the instance. To install the GCE SDK:

$ curl https://sdk.cloud.google.com | bash

To prepare the image bundle:

$ gcimagebundle -d /dev/sda -r / -o /tmp --loglevel=DEBUG --log\_file=/tmp/image\_bundle.log

Now the last steps will be upload the bundle and serve it.

You'll need to authenticate yourself with your GCE’s project (follow on-screen directions):

$ gcloud auth login

Create a storage bucket for your image (the name has to be unique across Google Cloud Storage globally). Mine is “my-gce-images”.  Let’s upload our bundle and instruct the GCE to build image out of it:

$ gsutil cp /tmp/<whatEverNameCreatedByBundelTool> gs://<your-gce-images-bucket>/<name-of-image>.tar.gz

$ gcutil addimage <image-name-to-be> gs://<your-gce-images-bucket>/<name-of-image>.tar.gz -project <project-ID>

That would be it and you should be able to see your new and shining Ubuntu image under “Images” in your GCE project.