Automating your Ubuntu Workstation Install

Formatting your drive and re-creating your Ubuntu workstation periodically (weekly) is an advantageous behaviour. Two important lean principles are

- if it hurts, do it more often
- (and) bring pain forward

So building your workstation weekly is a valuable practise.

Leave it and the task looms larger and larger and gets more and more daunting as time goes by.

Do it frequently and you ***absorb small shocks*** as you go along and gain much valued feedback on your behaviour in the past week. Now the rebuild is ***neither painful nor daunting*** as it's just one week's worth of creation activity to assimilate.

Create a Genesis User

Purpose - the genesis user does creation tasks - it is a no password sudoer - so you enter your password once and become the genesis user then let the scripts run whilst you take a break.

```
sudo adduser --home /var/opt/genesis --shell /bin/bash --gecos 'Genesis Creation User' genes
sudo install -d -m 755 -o genesis -g genesis /var/opt/genesis
```

The above simply create the user and there home directory within ***/var/opt***

Give an automaton User Sudo Priveleges

After modifying the user we actually need to be root otherwise a permission denied error for changing ***/etc/sudoers*** will ensue.

```
sudo usermod -a -G sudo genesis
sudo su root
sudo echo 'genesis ALL=NOPASSWD: ALL' >> /etc/sudoers
```

Now with a new terminal you become genesis and continue sudo su genesis

Ubuntu - the Big Upgrade

After installing on Ubuntu we need to do a ***big upgrade*** to get the latest versions of the key packages. Here it is.

```
sudo apt-get update && sudo apt-get --assume-yes upgrade
```

Install Git and Clone the Project

```
sudo apt-get install --assume-yes git ruby-full
git config --global core.autocrlf input
git config --global user.name "@[user|full.name]"
git config --global user.email "@[user|email.address]"
git config credential.helper store
git clone @[@[project|id]|git_url] @[@[project|id]|git_mirror]
```

The id of the ***project*** that is selected can be used to gain the git_url and the git_mirror from the key/value stores.

Then as and when - you can issue the common commands

```
git add .
git commit -am "Git repository comment"
git push -u origin master
git config --global push.default simple
git push
```

Sync Down Publicly Available S3 Documentation

We want to have the books and documentation pertinent to our ***environment***, ***tools***, ***systems*** and ***practices*** readily available from publicly available registered S3 buckets.

Let's install the AWS tools and sync them down.

```
sudo apt-get update && sudo apt-get --assume-yes upgrade
sudo apt-get install --assume-yes awscli
aws --version
```

Install Vagrant and VirtualBox

Our development workstation needs a bare metal installation of ***Vagrant and VirtualBox*** for maximum performance in order to emulate real ***container management systems***.

```
sudo apt-get update && sudo apt-get --assume-yes upgrade
sudo apt-get install --assume-yes vagrant virtualbox virtualbox-dkms linux-headers-$(uname -
vagrant --version
sudo dpkg-reconfigure virtualbox-dkms
sudo dpkg-reconfigure virtualbox
VBoxManage --version
```

If ***codeVBoxManage –version*** spits out the below error, check that ***virtualbox-dkms*** is installed and you have run the two ***dpkg-reconfigure*** commands.

> WARNING: The character device /dev/vboxdrv does not exist. Please install the virtualbox-dkms package and the appropriate headers, most likely linux-headers-generic.

Vagrant Install and Secure Boot Warning

Vagrant and Secure Boot cannot exist happily side by side. One needs 3rd party drivers whilst the other disallows their use. If you must have both then a password is requested which you will have to then re-enter when rebooting the system.

The Secure Boot error says »

> Your system has UEFI Secure Boot enabled. UEFI Secure Boot is not compatible with the use of third-party drivers. The system will assist you in toggling UEFI Secure Boot. To ensure that this change is being made by you as an authorized user, and not by an attacker, you must choose a password now and then use the same password after reboot to confirm the change. If you choose to proceed but do not confirm the password upon reboot, Ubuntu will still be able to boot on your system but the Secure Boot state will not be changed. If Secure Boot remains enabled on your system, your system may still boot but any hardware that requires third-party drivers to work correctly may not be usable.

Rebooting after Secure Boot Warning

If you chose to bypass Secure Boot you must reboot your system.

You will then be asked for character 7 then 8 then ... etc of the password you gave. Once that is entered you can opt to switch off SecureBoot whilst you use Vagrant.

The VT-x is Disabled in the BIOS Issue

This error occurs only when Vagrant is actually using a Hypervisor (like VirtualBox) to create the Virtual Machine.

See build business websites for a discussion on this error.

Once you've gone into the BIOS and reset the setting - you should then be able to use Vagrant.

Verifying the Vagrant Install

Once Vagrant is provisioned use ***vagrant –version*** to verify the install.

Install Python Tools

```
sudo apt-get install --assume-yes python-pip
pip --version
pip install awscli --upgrade --user
sudo pip install awscli --upgrade pip
```

Mount the USB Drive Decrypt Keys for Usage

In order to perform the automated workstation build we need to have access to an auth provider. This could be key pass, vaults and/or other services but we need to authenticate with the auth provider to.

Using a buddy system whence two or three office members authenticate the provision of tokens on a USB drive that are valid for a specific time and fit for a given purpose.

Bucket Name is ***devops.books***

This policy created. ***policy.read-write.s3-bucket.devops.books***

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::@[@[project|id]|s3.bucket.name]"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
```

```
        "s3:GetObject",
        "s3:DeleteObject"
      ],
      "Resource": ["arn:aws:s3:::@[@[project|id]|s3.bucket.name]/*"]
    }
  ]}
```

Test AWS S3 Bucket User

Run the iam.users eco-system

Commands

Check that it can list the names of all buckets - that is a pre-requisite to the one-bucket read-write usage. Test that the ONLY bucket it can list is devops.books aws s3 ls s3://wiija.documents

An error occurred (AccessDenied) when calling the ListObjects operation: Access Denied

aws s3 ls s3://devops.books aws s3 cp /media/apollo/6464-31A4/BitbucketServer_0410.pdf s3://devops.books/bitbucket.server.manual.pdf upload: ../../media/apollo/6464-31A4/BitbucketServer_0410.pdf to s3://devops.books/bitbucket.server.manual.pdf

aws s3 ls s3://devops.books 2017-11-30 18:45:53 10894058 bitbucket.server.manual.pdf

```
sudo fdisk -l
sudo mkdir /media/usb.key.1
sudo mount /dev
```

Install Pandoc to Convert Markdown to PDF

```
sudo apt-get install --assume-yes pandoc
sudo apt-get install --assume-yes texlive-latex-base
sudo apt-get install --assume-yes texlive-fonts-recommended
sudo apt-get install --assume-yes texlive-fonts-extra
sudo apt-get install --assume-yes texlive-latex-extra
pandoc markdown-file.md -s -o markdown-new.pdf
```

Then you can run pandoc commands like

Remap (Launcher) Alt-Key in Ubuntu's Desktop for Emacs

If Ubuntu starts up the launcher when you press the ALT key in emacs - it pays to remap it.

- click *__System Settings__ (top right on/off switch)

- in ***Hardware*** tab choose ***keyboard***
- click the ***Shortcuts*** tab and ***Launchers***
- click on ***Key to Show the HUD***
- press the right ALT Gr button and is says ***Level3 Shift***

That's it - no more launcher activation when the ALT key is pressed,

Install Google Chrome on Ubuntu Workstation

Firefox comes pre-installed but Chrome does not. Installing it requires the below commands.

```
wget -q -O - https://dl-ssl.google.com/linux/linux_signing_key.pub | sudo apt-key add -
sudo sh -c 'echo "deb https://dl.google.com/linux/chrome/deb/ stable main" >> /etc/apt/sourc
sudo apt-get update
sudo apt-get install --assume-yes google-chrome-stable
```

Create Workstation Key

This is required if the workstation may be part of a cluster. For the other machines (or the local one) to communicate over SSH we need to setup a key.

Everything is described in detail below.

[[https://www.build-business-websites.co.uk/ruby-ssh-windows-to-linux-net-ssh/]]

We needed to have installed ***openssh-server*** with apt-get.

@todo - We Still Need to Automate the below Command.

```
ssh-keygen -t rsa -C "Workstation Key"
```

Copy public key to the ***~/.ssh/authorized_keys*** file.

sudo cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys sudo chown *USER* :USER $HOME/.ssh/authorized_keys sudo chmod 400 $HOME/.ssh/authorized_keys cp $HOME/.ssh/id_rsa /media/apollo/6464-31A4/workstation.brain-pad.key.pem