**Week-1**

Installation and Introduction to Virtual Box

1. Install virtual box.
2. Create and run Virtual machines on Open source Operating systems.

# Ans-stall VirtualBox on Ubuntu

The objective of this Lab experiment is is to install VirtualBox on [Ubuntu 20.04](https://linuxconfig.org/ubuntu-20-04-guide) Focal Fossa Linux. VirtualBox is a free and open-source hosted hypervisor for x86 virtualization developed and maintained by the Oracle Corporation.

## Software Requirements and Conventions Used

| *Software Requirements and Linux Command Line Conventions* | |
| --- | --- |
| **Category** | **Requirements, Conventions or Software Version Used** |
| System | [Installed Ubuntu 20.04](https://linuxconfig.org/how-to-install-ubuntu-20-04-focal-fossa-desktop) or [upgraded Ubuntu 20.04 Focal Fossa](https://linuxconfig.org/how-to-upgrade-ubuntu-to-20-04-lts-focal-fossa) |
| Software | Oracle Virtualbox |
| Other | Privileged access to your Linux system as root or via the **sudo** command. |
| Conventions | **#** – requires given [linux commands](https://linuxconfig.org/linux-commands" \t "_blank) to be executed with root  privileges either directly as a root user or by use of **sudo** command **$** – requires given [linux commands](https://linuxconfig.org/linux-commands" \t "_blank) to be executed as a regular  non-privileged user |

## Install VirtualBox on Ubuntu 20.04 step by step instructions

### Install VirtualBox from command line

To install virtualbox from a command line [open the terminal](https://linuxconfig.org/shortcuts-to-access-terminal-on-ubuntu-20-04-focal-fossa) window and enter the following **apt** command:

$ sudo apt install virtualbox

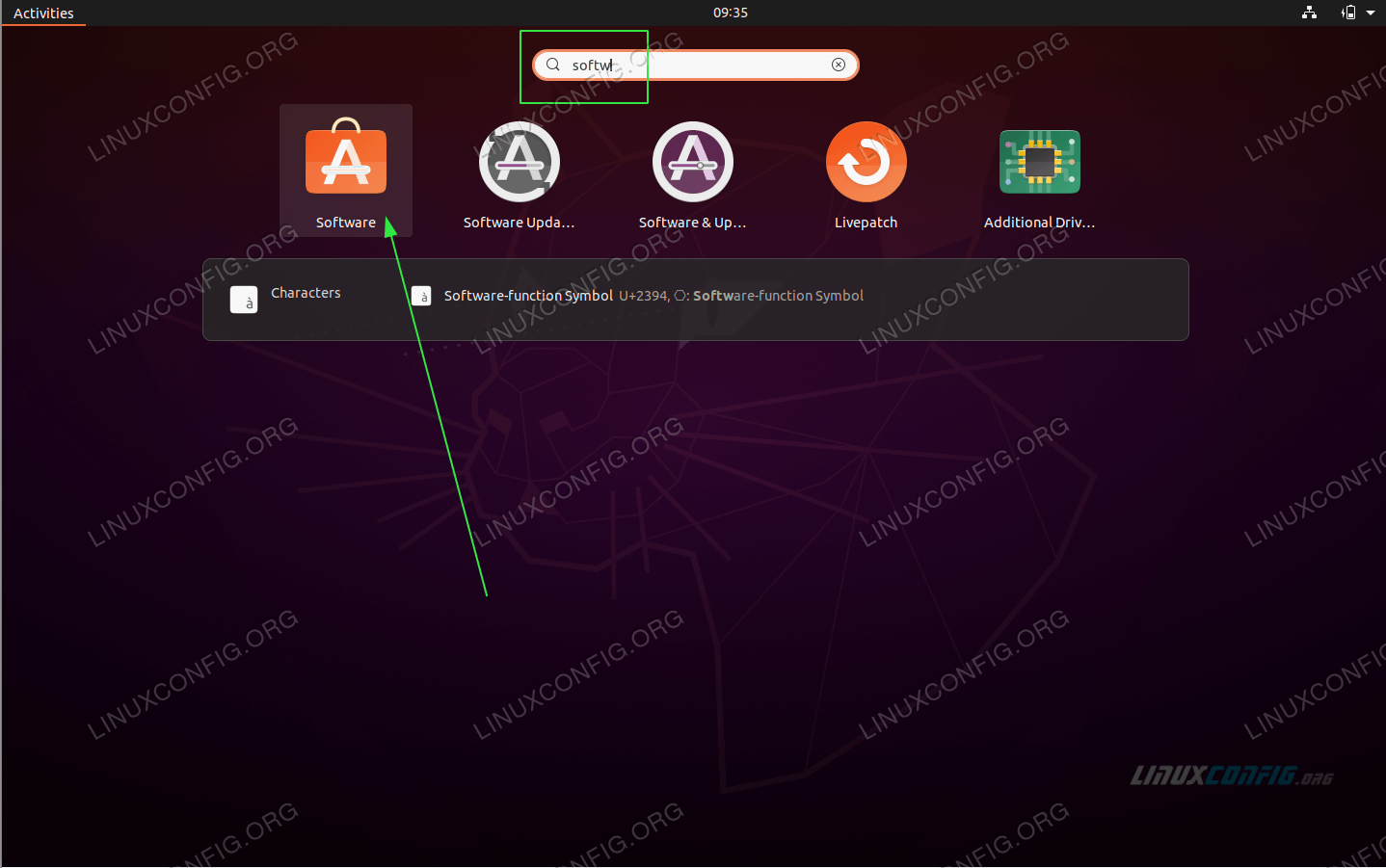
All done. To start the Virtualbox execute:

$ virtualbox

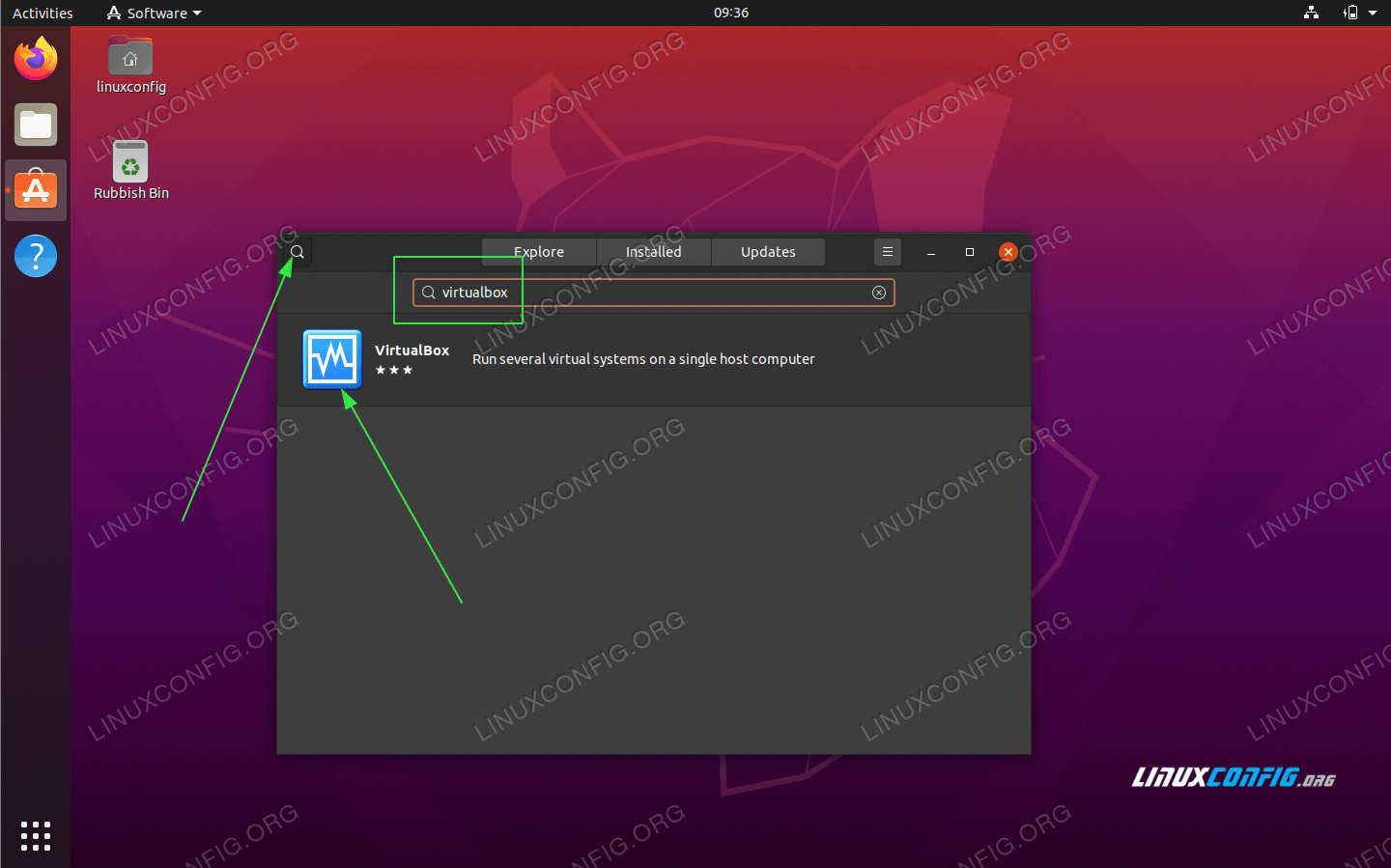
You can now for example test the VirtualBox installation by creating a new Ubuntu 64-bit virtual machine and booting into [Ubuntu 20.04 downloaded](https://linuxconfig.org/ubuntu-20-04-download) ISO image.

Optionally you can now [install VirtualBox Extension Pack](https://linuxconfig.org/virtualbox-extension-pack-installation-on-ubuntu-20-04-focal-fossa-linux) to add some extra features to your VirtualBox installation.

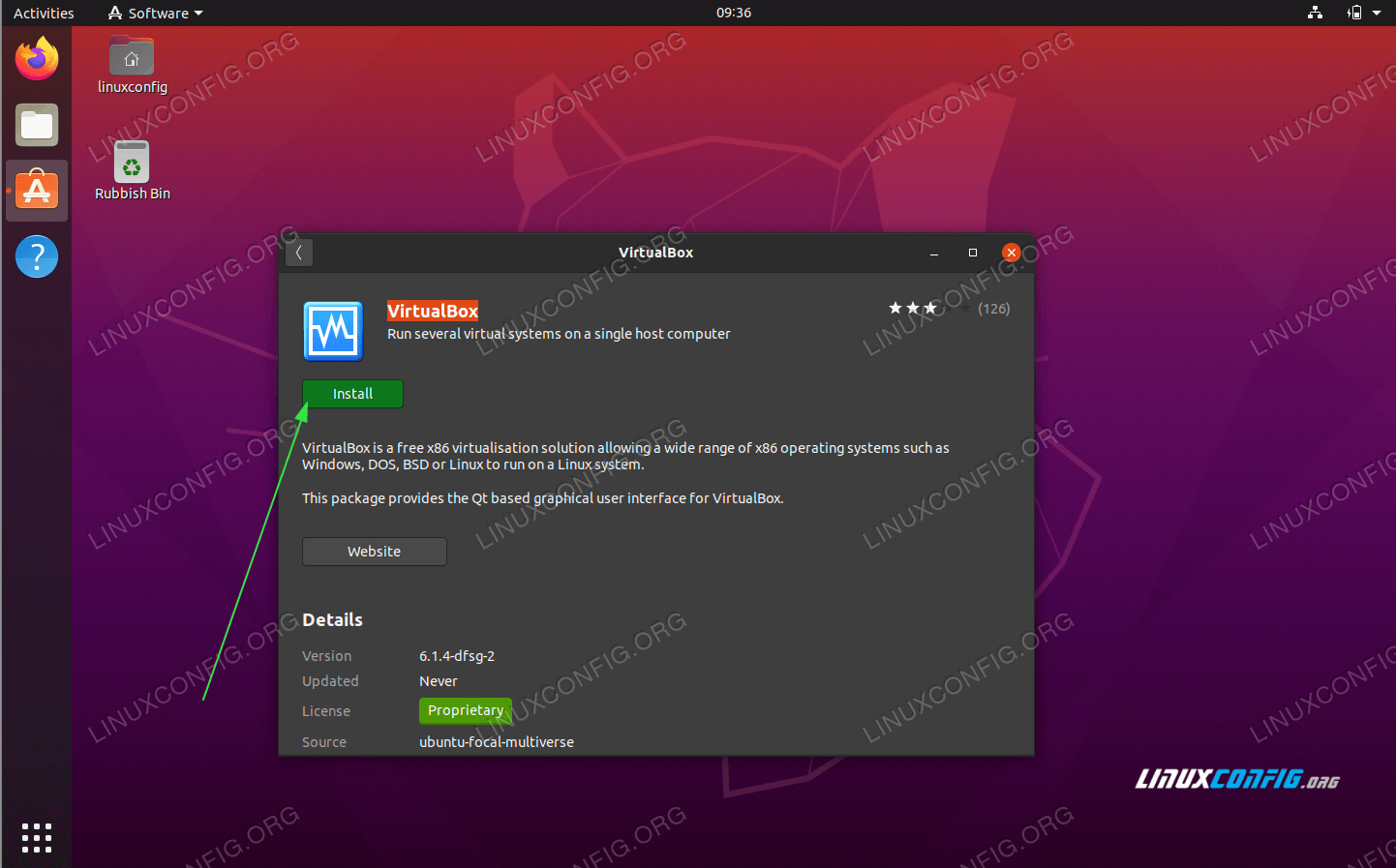
### Install VirtualBox from GNOME Desktop

[](https://linuxconfig.org/wp-content/uploads/2020/03/01-install-virtualbox-on-ubuntu-20-04-focal-fossa-linux.png)

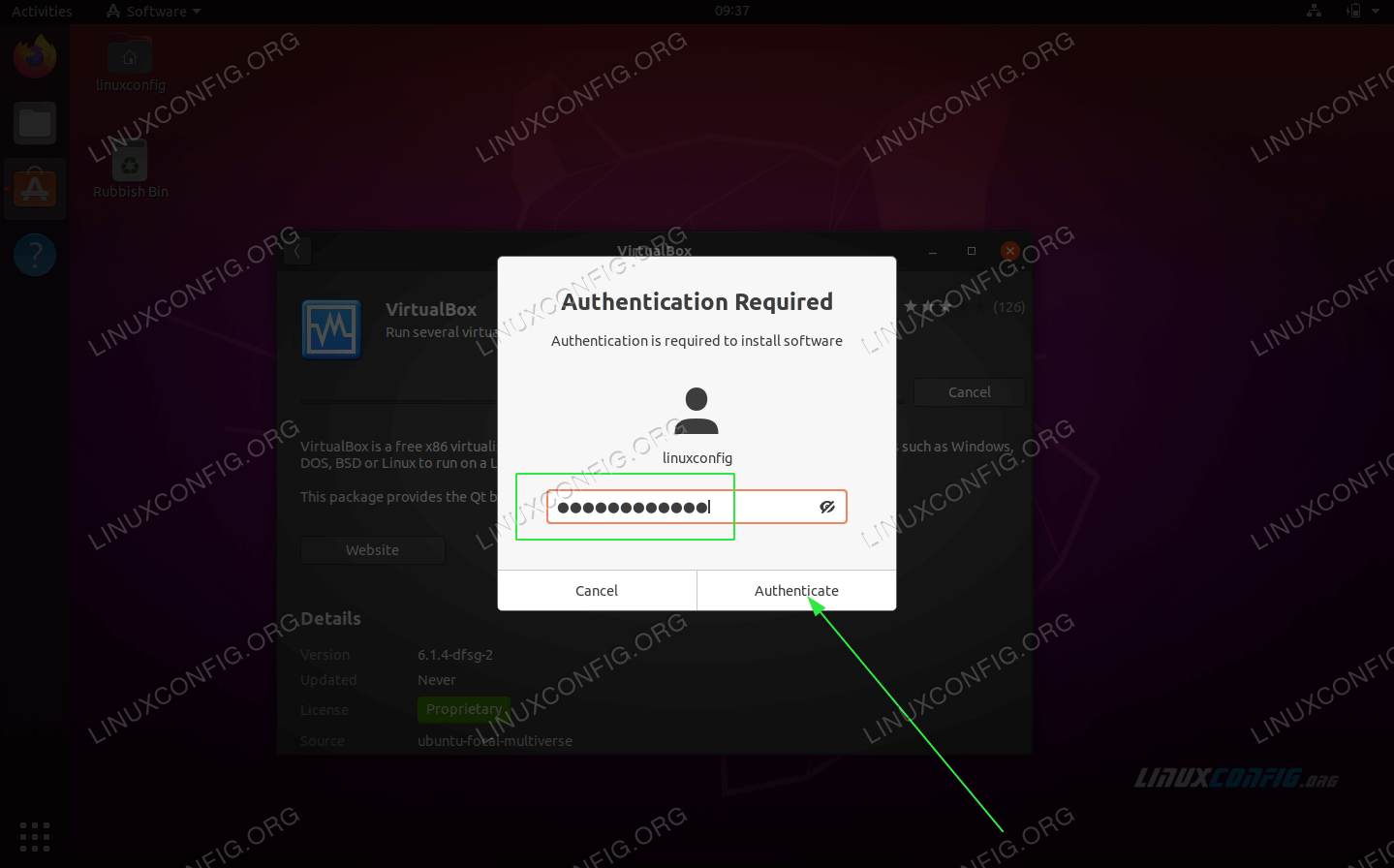
*Use the top left****Activities****menu to open the****Software****application.*

[](https://linuxconfig.org/wp-content/uploads/2020/03/02-install-virtualbox-on-ubuntu-20-04-focal-fossa-linux.png)

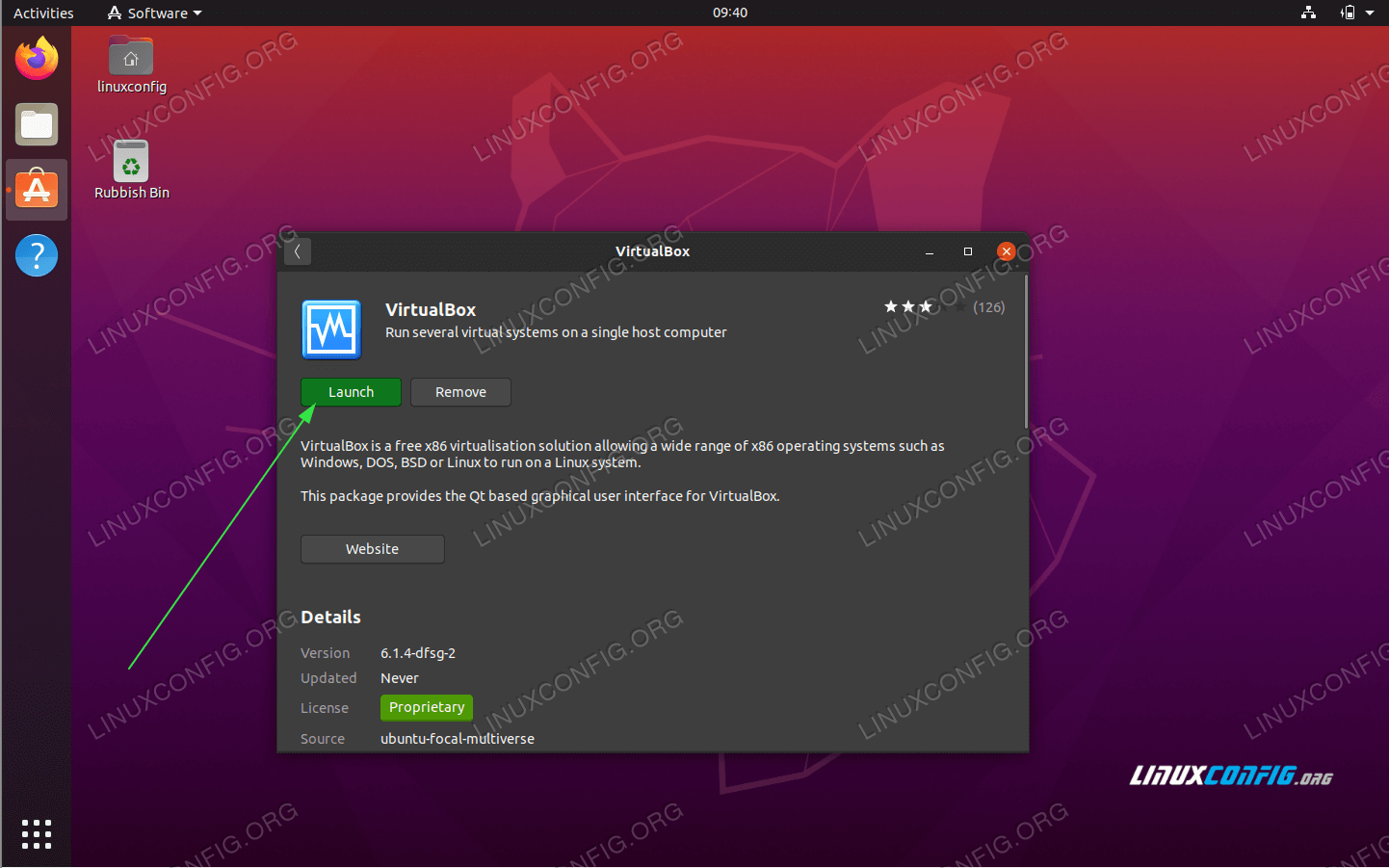
*Search for****virtualbox****keyword.*

[](https://linuxconfig.org/wp-content/uploads/2020/03/03-install-virtualbox-on-ubuntu-20-04-focal-fossa-linux.png)

*To begin the VirtualBox installation hit the****install****button.*

[](https://linuxconfig.org/wp-content/uploads/2020/03/04-install-virtualbox-on-ubuntu-20-04-focal-fossa-linux.png)

*Enter your password. Your user must belong to the [sudo group](https://linuxconfig.org/how-to-create-sudo-user-on-ubuntu-20-04-focal-fossa-linux" \t "_blank) in order to proceed with the installation.*

[](https://linuxconfig.org/wp-content/uploads/2020/03/05-install-virtualbox-on-ubuntu-20-04-focal-fossa-linux.png)

*Once the VirtualBox installation is completed simply click on****Launch****button or use****Activities****menu to start the VirtualBox application.*

**Week-2**

Installing Cloud Foundry in a local host and exploring CF commands

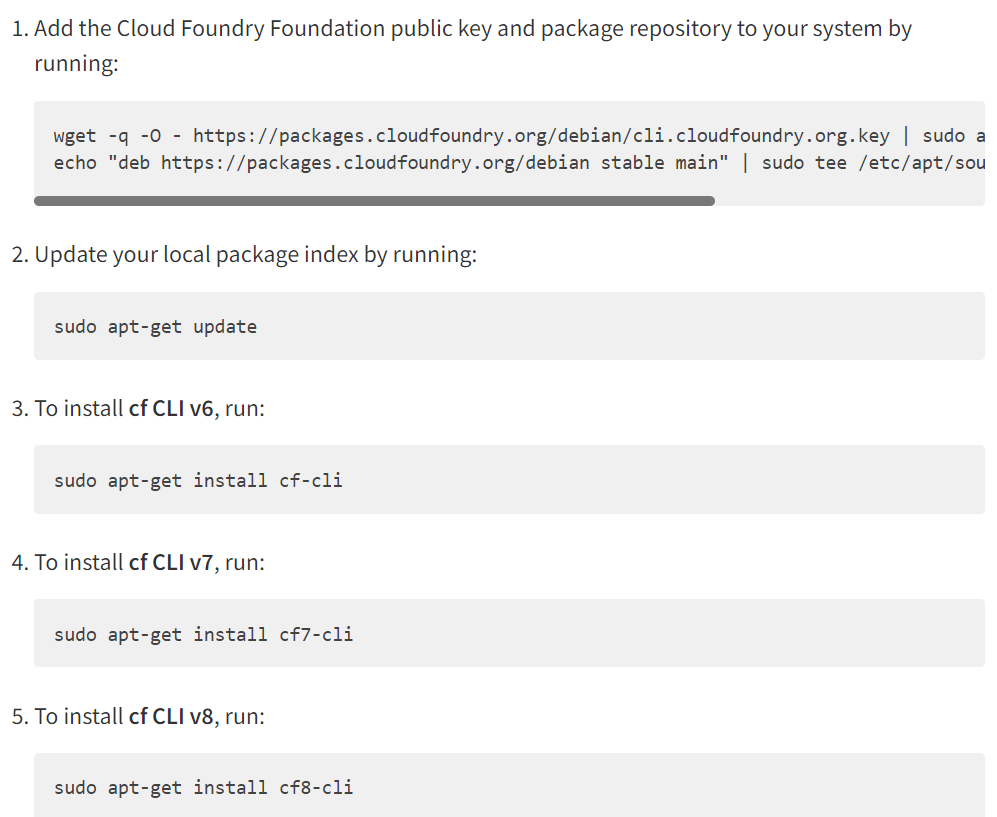
1. Learn about Cloud Foundry installation.
2. Implement Cloud services in it.

Install the cf CLI Using a Package Manager

* These sections describe how to install the cf CLI using a package manager. You can install the cf CLI using a package manager on Mac OS X and Linux operating systems.

**Linux Installation**

* There are two ways to install the cf CLI using a package manager, depending on your Linux distribution.
* To install the cf CLI on Debian and Ubuntu-based Linux distributions:



****

**Uninstall the cf CLI**

Installer

* If you installed the cf CLI with an installer, follow the procedure in this section that is specific to your operating system.
* To uninstall the cf CLI on Mac OS X:
* Delete the binary /usr/local/bin/cf.
* Delete the directory /usr/local/share/doc/cf-cli.

To uninstall the cf CLI on Windows:

* Navigate to the Control Panel and click Programs and Features.
* Select Cloud Foundry CLI VERSION.
* Click Uninstall.

**Use a Package Manager**

**Windows Installation**

* You can run cf CLI in either the Windows Subsystem for Linux (WSL), also known as Bash on Windows, or in the Windows command line.
* To use WSL, follow the Linux Installation instructions.
* To use the cf CLI installer for the Windows command line, perform the following steps:

1. Download the Windows installer.
2. Unpack the zip file.
3. Right click on the ‘cf\_installer’ executable and select “Run as Administrator”
4. When prompted, click Install, then Finish.
5. To verify your installation, open a command prompt and type cf. If your installation was successful, the cf CLI help listing appears. You may need to restart the command prompt to see the cf CLI help listing appear.

**Week-3**

Implementing Security and Networking in Cloud Foundry.

1. Create security groups in CF.
2. Manage container security and network traffic rules.

* **Introduction**

Application security groups act as virtual firewalls to control outbound traffic from the applications in your deployment. A security group consists of a list of network egress access rules.

An administrator can assign one or more security groups to a Cloud Foundry deployment or to a specific [space](https://docs.huihoo.com/cloudfoundry/documentation/concepts/roles.html#spaces) in an [org](https://docs.huihoo.com/cloudfoundry/documentation/concepts/roles.html#orgs) within a deployment.

**Note**: A security group assigned to a Cloud Foundry deployment affects all spaces in all orgs within the deployment.

Within a space, Cloud Foundry runs each instance of an application inside a separate application container. When you launch an application for the first time, Cloud Foundry creates a new container for each application instance, then applies any space-specific and deployment-wide security groups to the container. Cloud Foundry determines whether to allow or deny outbound traffic from the container by evaluating the rules defined in these security groups.

* **Create security groups in CF.**

A security group consists of a list of operator-defined network egress allow rules. These rules define the outgoing traffic allowed to application containers. Each rule contains the following three parts:

* **Protocol**: TCP, UDP, or ICMP
* **Open Port / Port Range**:
  + For TCP and UDP: Either a single port or a range of ports
  + For ICMP: An ICMP type and code
* **Destination**: Destination of the traffic allowed by this rule as an IP address or CIDR block

Run cf create-security-group SECURITY-GROUP PATH-TO-RULES-FILE from a command line to create a security group named SECURITY-GROUP. PATH-TO-RULES-FILE can be an absolute or relative path to a rules file. The rules file must be a JSON-formatted single array containing objects that describe the rules.

Example JSON-formatted rules file:

[{"protocol":"tcp","destination":"10.0.11.0/24","ports":"1-65535"},

{"protocol":"udp","destination":"10.0.11.0/24","ports":"1-65535"}]

* **Binding security groups in CF.**

A security group consists of a list of rules. You must bind this list to either your entire deployment or to a space in an org for Cloud Foundry to apply the rules to outgoing traffic.

**Note**: New security rules apply to new containers as they are created, but not to containers that are already running when the rules are created. To apply new security rules immediately, restart all running apps in the relevant space, or across your whole installation if the new rules are for all spaces.

To apply the rules in a security group to your entire Cloud Foundry deployment, bind the security group to either the **Default Staging** or the **Default Running** security group set. Cloud Foundry applies the rules in every security group in the Default Staging and Default Running sets to all applications in your Cloud Foundry deployment.

**Note**: If you do not bind a security group to either the [Default Staging](https://docs.huihoo.com/cloudfoundry/documentation/adminguide/app-sec-groups.html#bind-org) or the [Default Running](https://docs.huihoo.com/cloudfoundry/documentation/adminguide/app-sec-groups.html#bind-org) set, Cloud Foundry only applies the rules to applications created in the specific space where you bound the security group.

**Binding to your Cloud Foundry Deployment**

To create a rule to be applied to every space in every org of your Cloud Foundry deployment, bind the security group to either the **Default Staging** or the **Default Running** security group set.

Cloud Foundry applies the rules in every security group in these sets as follows:

* **Default Staging**: Cloud Foundry applies rules in this set to every application staged anywhere in your CF deployment. To bind a security group to the Default Staging set, run cf bind-staging-security-group SECURITY-GROUP.
* **Default Running**: Cloud Foundry applies rules in this set to every application running anywhere in your CF deployment. To bind a security group to the Default Running set, run cf bind-running-security-group SECURITY-GROUP.
* **Binding to a Space**

Run cf bind-security-group SECURITY-GROUP ORG SPACE to bind a security group to a specific space. Cloud Foundry applies the rules that this security group defines to all application containers in the space. A space may belong to more than one security group.

**Network Traffic Rules Evaluation Sequence**

Cloud Foundry evaluates security groups and other network traffic rules in a strict priority order. Cloud Foundry returns an allow, deny, or reject result for the first rule that matches the outbound traffic request parameters, and does not evaluate any lower-priority rules.

Cloud Foundry evaluates the network traffic rules for an application in the following order:

1. **Security Groups**: The rules described by the Default Staging set, the Default Running set, and all security groups bound to the space.
2. **Warden ALLOW rules**: Any Warden Server configuration allow rules. Set Warden Server configuration rules in the Droplet Execution Agent (DEA) configuration section of your deployment manifest.
3. **Warden DENY rules**: Any Warden Server configuration deny rules. Set Warden Server configuration rules in the DEA configuration section of your deployment manifest.
4. **Hard-coded REJECT rule**: Cloud Foundry returns a reject result for all outbound traffic from a container if not allowed by a higher-priority rule.

**Viewing Security Groups**

Run the following commands to view information about existing security groups:

* cf security-groups: Displays all security groups in an org
* cf staging-security-groups: Displays all security groups in the Default Staging set
* cf running-security-groups: Displays all security groups in the Default Running set
* cf security-group SECURITY-GROUP: Displays details about the specified security group

**Week-4**

Installation and Understanding features of IaaS using Open stack

1. Install Openstack.
2. Implement IaaS in openstack.

**a)** Install Openstack.

**Steps to Install Openstack on Ubuntu with Devstack**

Installing OpenStack on Ubuntu is a rather complex process. But it is made easy by Devstack. The steps to install it, are quite easy even if you’re not much proficient with the command line, simply follow the steps and get it up and running.

**Step 1: Preparing the system**

Before we start off, we need to ensure that our system is updated, for that run following command:

|  |  |
| --- | --- |
|  | sudo apt-get update && sudo apt-get upgrade -y |

The command will ask for root privileges. Enter your user password and wait for your system to upgrade. After the upgrade is finished, make sure to **reboot** your system. It will initialize and setup your upgrades in the next reboot.

**Step 2: Creating *stack* user with Sudo privileges**

Now, it’s time to start with the important steps to install Openstack on Ubuntu.We will first create a new user named *stack* for our system to setup OpenStack, as it should be installed on a non-root user with sudo enabled.

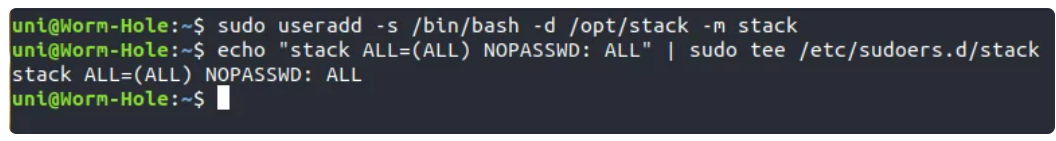
Open a fresh terminal, and run the useradd command:

|  |  |
| --- | --- |
|  | sudo useradd -s /bin/bash -d /opt/stack -m stack |

You also need to enable*stack* user to have root privileges and run without a password, for that run:

|  |  |
| --- | --- |
|  | echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack |

The Output will look like this –

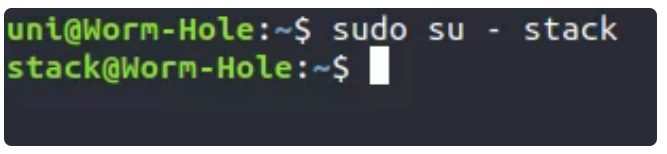


Adding User – “stack”

Once you have created the *stack* user, it’s time to log in to it using the following command:

|  |  |
| --- | --- |
|  | sudo su - stack |

It will log in you to bash, as stack user.



Logging-in as User “stack”

**Step 3: Downloading Devstack**

For this step, we considered you already git installed on your system. Now, enter this command to download/clone devstack from its repository to your system:

|  |  |
| --- | --- |
|  | git clone https://opendev.org/openstack/devstack |

Devstack repo contains a script *stack.sh* , which we will use to setup OpenStack. It also contains templates for configuration files.

**Step 4: Creating configuration (.conf) file for Devstack**

Now, we have downloaded DevStack and need to setup our configuration files for it.You need to first navigate to the devstack folder, by running:

|  |  |
| --- | --- |
|  | cd devstack |

Afterwards, create a local.conf file, by running:

|  |  |
| --- | --- |
|  | vim local.conf |

and paste the following content –

|  |  |
| --- | --- |
|  | [[local|localr]]    ADMIN\_PASSWORD=StrongAdminSecret  DATABASE\_PASSWORD=$ADMIN\_PASSWOCinder  RABBIT\_PASSWORD=$ADMIN\_PASSWORD  SERVICE\_PASSWORD=$ADMIN\_PASSWORD |

If you are not much familiar with Vim, you can read through the [vim tutorial](https://www.linuxfordevices.com/tutorials/linux/vim-tutorial). For now, you can just paste using the mouse by right-clicking and clicking Paste and enter **:x** to save & exit. Here, I used a minimal configuration to setup DevStack, you can explore

**Note:**

1. *StrongAdminSecret* is the password we used here, you can change it with your choice.

2. You can find a sample configuration file for*local.conf* in the Samples directory under the Devstack repository.

**Step 5: Installing Openstack with Devstack**

Now, as we have setup the configuration files properly.

Let’s run the script to setup OpenStack on our system, using the following command:

|  |  |
| --- | --- |
|  | ./stack.sh |

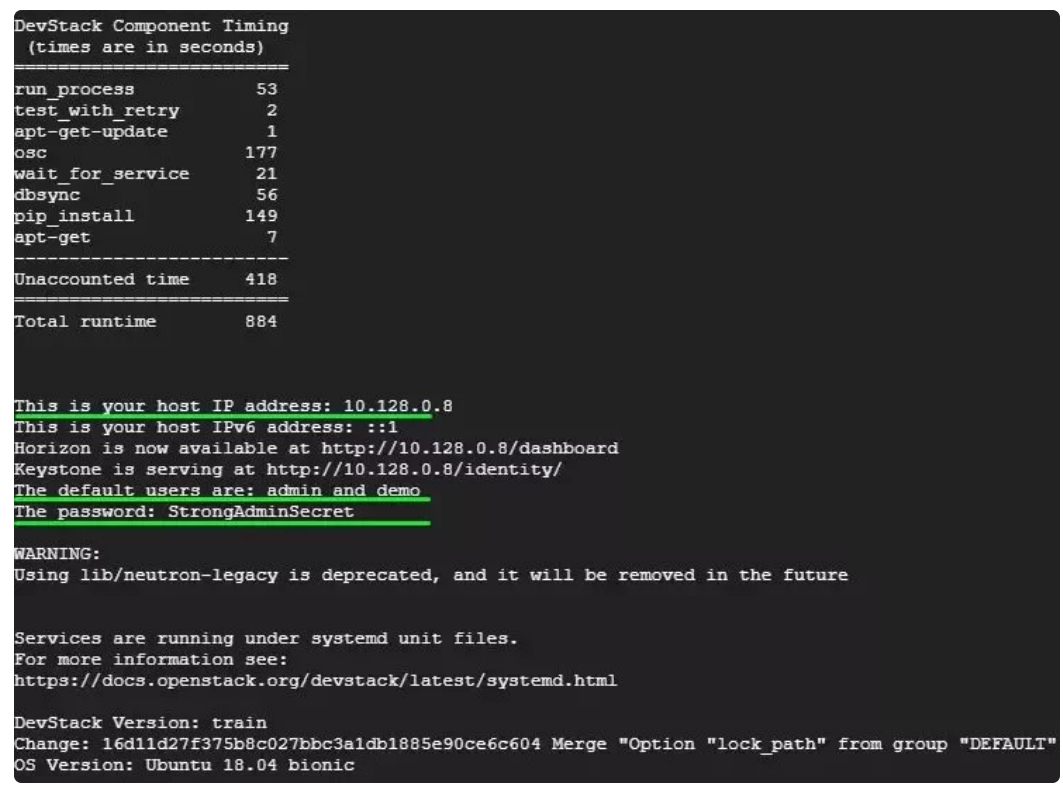
(This script we’re using is part of DevStack itself)

The script will install the listed features for your OpenStack environment –

* Horizon – OpenStack Dashboard
* Keystone – Identity Service
* Nova – Compute Service
* Glance – Image Service
* Neutron – Network Service
* Placement – Placement API
* Cinder – Block Storage Service

The setup will take around 10 to 20 minutes, based on your system performance and internet speed, as many git trees and packages are installed during the process.

After your installation successfully finishes, your terminal will look like the image below.



Devstack Installed

Now, we can see that it is saying that Horizon (Openstack Dashboard) is available at the given URL, it will vary from system to system.

**Step 6: Accessing OpenStack using a web browser**

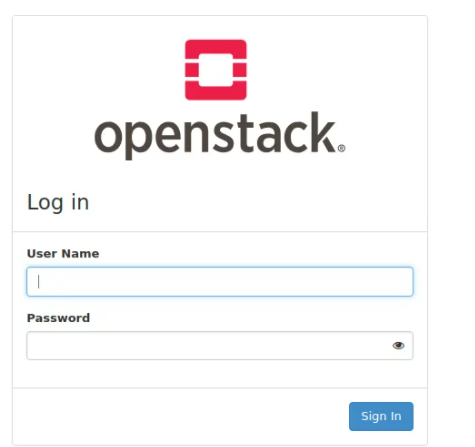
Now, as we have successfully setup OpenStack using Devstack, let’s access it via our browser.

Browse this URL on your browser –

|  |  |
| --- | --- |
| 1 | https://server-ip/dashboard |

Or try

|  |  |
| --- | --- |
| 1 | https://localhost/dashboard |



b) Implement IaaS in openstack.

**Deployment of OpenStack using DevStack**

DevStack is used to quickly create an OpenStack development environment. It is also used to demonstrate the starting and running of OpenStack services, and provide examples of using them from the command line. DevStack has evolved to support a large number of Configuration options and alternative platforms and support services. It can be considered as the set of scripts which install all the essential OpenStack services in the computer without any additional software or configuration. To implement DevStack, first download all the essential packages, pull in the OpenStack code from various OpenStack projects, and set everything for the deployment.

To install OpenStack using DevStack, any Linux-based distribution with 2GB RAM can be used to start the implementation of IaaS.  
Here are the steps that need to be followed for the installation.

**1. Install Git**

|  |
| --- |
| $ sudo apt-get install git |

**2. Clone the DevStack repository and change the directory. The code will set up the cloud infrastructure.**

|  |
| --- |
| $ git clone http://github.com/openstack-dev/devstack  $ cd devstack/    /devstack$ ls |

**3. Execute the stack.sh script:**

|  |
| --- |
| /devstack$ ./stack.sh |

Here, the MySQL database password is entered. There is no need to worry about the installation of MySQL separately on this system. We have to specify a password and this script will install MySQL, and use this password there.  
Finally, we will have the script ending as follows:

|  |
| --- |
| + merge\_config\_group /home/r/devstack/local.conf post-extra  + local localfile=/home/r/devstack/local.conf  + shift  + local matchgroups=post-extra  + [[ -r /home/r/devstack/local.conf ]]  + **return** 0  + [[ -x /home/r/devstack/local.sh ]]  + service\_check  + local service  + local failures  + SCREEN\_NAME=stack  + SERVICE\_DIR=/opt/stack/status  + [[ ! -d /opt/stack/status/stack ]]  ++ ls /opt/stack/status/stack/\*.failure  ++ /bin/true  + failures=  + [ -n  ]  + set +o xtrace |

* Horizon is now available at *http://1.1.1.1/*
* Keystone is ser
* ving at *http://1.1.1.1:5000/v2.0/*
* Examples on using the *novaclient* command line are in *exercise.sh*
* The default users are: admin and demo
* The password: nova
* This is your host IP: *1.1.1.1*

After all these steps, the machine becomes the cloud service providing platform. Here, 1.1.1.1 is the IP of my first network interface.  
We can type the host IP provided by the script into a browser, in order to access the dashboard Horizon. We can log in with the username admin or demo and the password admin.  
You can view all the process logs inside the screen, by typing the following command:

|  |
| --- |
| $ screen -x |

Executing the following will kill all the services, but it should be noted that it will not delete any of the code.  
To bring down all the services manually, type:

|  |
| --- |
| $ sudo killall screen |

|  |
| --- |
|  |

**Cinder on DevStack**

Cinder is a block storage service for OpenStack that is designed to allow the use of a reference implementation (LVM) to present storage resources to end users that can be consumed by the OpenStack Compute Project (Nova). Cinder is used to virtualise the pools of block storage devices. It delivers end users with a self-service API to request and use the resources, without requiring any specific complex knowledge of the location and configuration of the storage where it is actually deployed.

All the Cinder operations can be performed via any of the following:

1. CLI (Cinders *python-cinderclient* command line module)  
2. GUI (Using OpenStacks GUI project*horizon*)  
3. Direct calling of Cinder APIs

**Creation and deletion of volumes:** To create a 1 GB Cinder volume with no name, run the following command:

|  |
| --- |
| $ cinder create 1 |

To see more information about the command, just type cinder help *<command>*

|  |
| --- |
| $ cinder help create    usage: cinder create [--snapshot-id <snapshot-id>]  [--source-volid <source-volid>] [--image-id <image-id>]  [--display-name <display-name>]  [--display-description <display-description>]  [--volume-type <volume-type>]  [--availability-zone <availability-zone>]  [--metadata [<key=value> [<key=value> ...]]]  <size>  Add a new volume.  Positional arguments:  <size> Size of volume **in** GB  Optional arguments:  --snapshot-id <snapshot-id>  Create volume from snapshot id (Optional,  Default=None)  --source-volid <source-volid>  Create volume from volume id (Optional, Default=None)  --image-id <image-id>  Create volume from image id (Optional, Default=None)  --display-name <display-name>  Volume name (Optional, Default=None)  --display-description <display-description>  Volume description (Optional, Default=None)  --volume-type <volume-type>  Volume type (Optional, Default=None)  --availability-zone <availability-zone>  Availability zone **for** volume (Optional, Default=None)  --metadata [<key=value> [<key=value> ...]]  Metadata key=value pairs (Optional, Default=None) |

To create a Cinder volume of size 1GB with a name, using *cinder create –display-name myvolume:*

|  |
| --- |
| $ cinder create --display-name myvolume 1  +------------------+----------------------------------------+  | Property | Value |  +------------------+-------------------------------------- -+  | attachments | [] |  | availability\_zone | nova |  | bootable | false |  | created\_at | time | | display\_description | None |  | display\_name | myvolume | | id | id |  | metadata | {} |  | size | 1 |  | snapshot\_id | None |  | source\_volid | None |  | status | creating |  | volume\_type | None |  +-----------------------+----------------------------------+ |

To list all the Cinder volumes, using *cinder list:*

|  |
| --- |
| $ cinder list |
| ID Status Display Name Size Volume type Bootable Attached To  id1 Available Myvolume 1 None False  id2 Available None 1 None False |

To delete the first volume (the one without a name), use the *cinder delete <volume\_id>* command. If we execute*cinder* list really quickly, the status of the volume going to deleting can be seen, and after some time, the volume will be deleted:

|  |
| --- |
| $ cinder delete id2    $ cinder list  ID Status Display Name Size Volume type Bootable Attached To  id1 Available Myvolume 1 None False  id2 Deleting None 1 None False |

Volume snapshots can be created as follows:

|  |
| --- |
| $ cinder snapshot-create id2  +---------------------+---------------------------------+  | Property | Value |  +---------------------+-----------------------------------+  | created\_at | TimeStamp |  | display\_description | None |  | display\_name | None |  | id | snapshot2 |  | metadata | {} |  | size | 1 |  | status | creating |  | volume\_id | id2 |  +---------------------+----------------------------------+ |

All the snapshots can be listed as follows:

|  |
| --- |
| $ cinder snapshot-list    ID Volume ID Status Display Name Size  Snapshotid1 id2 Available None 1 |

You can also create a new volume of 1GB from the snapshot, as follows:

|  |
| --- |
| $ cinder create --snapshot-id snapshotid1 1  +---------------------+--------------------------------------  | Property | Value  +---------------------+--------------------------------------  | attachments | [] |  | availability\_zone | nova |  | bootable | false |  | created\_at | creationtime |  | display\_description | None |  | display\_name | None |  | id | v1 |  | metadata | {} |  | size | 1 |  | snapshot\_id | snapshotid1 |  | source\_volid | None |  | status | creating |  | volume\_type | None |  +---------------------+-------------------------------------+ |

There are lots of functions and features available with OpenStack related to cloud deployment. Depending upon the type of implementation, including load balancing, energy optimisation, security and others, the cloud computing framework OpenStack can be explored a lot

**Week-5**

Installing and Understanding Eucalyptus cloud -I

1. Install Eucalyptus cloud.

INSTALLATION OF EUCALYPTUS

step 1:sudo apt install snapd

this command is only for 14,15 versions.

step 2: sudo snap install eucalyptus-tools

INSTALLATION OF CLOUD FOUNDRY:

Try command:

sudo snap install eucalyptus-console

for cf package installation try:sudo apt install confluence.

some cloud foundry commands:

1.cf -h or cf -help

- will get the verion,different commands,author.

link for other commands:

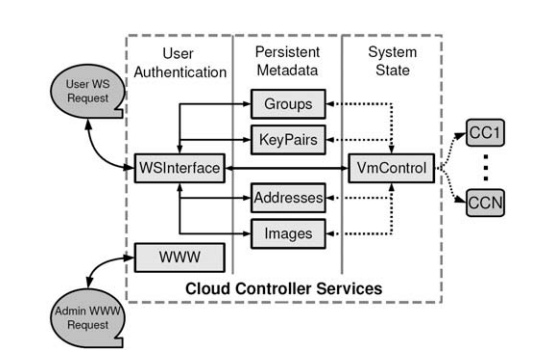
1. Understand the concepts of Cluster Controller, and Cloud Controller (CLC).

Eucalyptus is comprised of several components: Cloud Controller, Walrus, Cluster Controller, Storage Controller, and Node Controller. Each component is a stand-alone web service. This architecture allows Eucalyptus both to expose each web service as a well-defined, language-agnostic API, and to support existing web service standards for secure communication between its components.

#### Cloud Controller

The Cloud Controller – also known as CLC – is the highest level in Eucalyptus. There is one Cloud Controller per infrastructure. The Cloud Controller is in charge of the following tasks:

* Connect to virtual instances via SSH
* Provide a Front end for the Web Services that are EC2 and S3 compatible
* The Cloud Controller acts as a Meta Scheduler for the Cloud Infrastructure and determines which infrastructure to use.
* The Cloud Controller collects resource information from Cluster Controllers. The Cloud Controller runs per default on same machine as Walrus und the Storage Controller.
* The Cloud Controller (CLC) is the entry-point into the cloud for administrators, developers, project managers, and end-users. The CLC queries other components for information about resources, makes high-level scheduling decisions, and makes requests to the Cluster Controllers (CCs). As the interface to the management platform, the CLC is responsible for exposing and managing the underlying virtualized resources



**Cluster Controller**

The Cluster Controller (CC) comes next in hierarchy after the Cloud Controller (CLC). There is exactly one Cluster Controller per location. A location could be compared to an Availability Zone within a Region in Amazon Web Services. The Cluster Controller is basically in charge of receiving requests from the Cloud Controller to deploy new virtual Instances. The Cluster Controller decides which Node is used for the new virtual Instance. The Cluster Controller also maintains virtual Networks available to the instances and collects information about the Node Controllers registered. This information is reported to the Cloud Controller. Each Cluster can have exactly one Cluster Controller. When a new Instance is started, the Cloud Controller is instructed with the Image, Instance Type and Instance Number. The Cloud Controller looks up a Cluster Controller with enough available resources and selects one to start the instance. The Cloud Controller now itself looks up Node Controllers with enough resource availability and instructs the Node Controller to launch a new virtual Instance. If the Image requested is not available on the Node, the Node Controller looks up the Image by asking the Cloud Controller. The Cloud Controller now provides the Image via Walrus to the Node.