Steps to do for maas installation.

1. Boot by using server image. choose option install multiple servers using ubuntu MAAS.
2. this is normal installation like ubuntu server. just choose install new maas.
3. or you can just install ubuntu server and then install maas by
   1. sudo add-apt-repository ppa:maas/stable
   2. sudo apt-get update
   3. sudo apt-get install maas
4. after complete installation. check for ip address. change in etc/network/interface if required. It should have 2 eth0 and eth1. one connected to dhcp(for internet) and one connected to your private network(where all nodes will be attached) give static ip(ask admin for free ip range so it should not conflict with existing one). add gateway, dns-nameservers
5. reboot system check again by ifconfig if both eth0 and eth1 are up and proper ips are assigned.
6. go to browser from pc can ping maas host. enter url http://{ip\_address\_maas}/MAAS
7. maas dashboard will open otherwise check ip again check maas again.
8. it will show maas\_create\_user adminuser command run it on host. it will ask for username(root), password,email provide it.
9. now refresh browser page it will ask for credentials give as you given in adminuser. maas dashboard will open with NODES above round containing 0. that is showing connected nodes.
10. you can see orange warning above about no images. click on link or click on image tab. select proper images(linux 14.4 lts,64) and click import
11. refresh page it will show importing image. it will take time 10-15 min depending on net speed.
12. as import image is going on. go in clusters you will see default cluster. click on it it will open page. at bottom it will show networks unmanaged. delete it add new.
13. when you click on add new it will open page having many fields. give name(maas-eth1), choose interface (eth0/eth1 one which is connected to node not to internet)
14. enter dhcp ranges dynamic are mandatory as per suggested by admin. it may dhcp server ip give maas server ip. click save. it will back to cluster page.
15. till this images will be imported check it.
16. now you have to generate and add ssh-keys.
    1. **ssh-keygen -t rsa**
    2. **enter enter enter**
    3. **cat id\_rsa.pub (location will come above)**
    4. **copy that key. go to user(upper right corner preferences)**
    5. **add ssh-key pest that key**
17. **sudo apt-get install etherwake (to send magic packet to power up a node )**
18. make changes given here so etherwake will work with private network only [http://askubuntu.com/questions/504452…](http://askubuntu.com/questions/504452/node-in-maas-not-waking-up-on-lan)
19. edit /etc/maas/templates/power/ether\_wake.template.
    1. comment out wake-on-lan lines.
       1. #elif [ -x /usr/bin/wakeonlan ]
       2. #then
       3. # /usr/bin/wakeonlan $mac\_address
    2. and adjust the following line in

**/usr/sbin/etherwake $mac\_address** to: **sudo /usr/sbin/etherwake -i {eth#} $mac\_address**

replace **eth#** by internal network eth port (eg eth0,eth1)

* 1. **sudo nano /etc/sudoers.d/99-maas-sudoers**
  2. **add into that file**
  3. **maas ALL= NOPASSWD: /usr/sbin/etherwake**

1. in dns-nameservers add maas server ip. reboot. test( nslookup nodename.maas ).
2. **sudo** **dpkg-reconfigure maas-cluster-controller** it will open GUI showing url with internet ip change it to internal (optional).
3. Preset node
   1. requirements for machines
   2. turn PXE boot on
   3. set virtualisation on
   4. move network at top of boot options
4. turn on node look if its booting from DHCP PXE otherwise check settings or network
5. wait for boot complete
6. when boot complete look if maas is detected node(yellow dot).
   1. click on edit node for detected node in nodes tab
   2. change name select power option as wake\_on\_lan
   3. add mac address
   4. save
7. click on commision node. status will go commissioning and then ready.
8. enable net forwording
   1. do **nslookup google**
   2. do **nslookup {nodename}.maas**
      1. if it’s unable to lookup add dns-nameservers {8.8.8.8} {maas ip address} to the /etc/network/interfaces

If you haven't already enabled forwarding in the kernel, do so.

* Open /etc/sysctl.conf and uncomment net.ipv4.ip\_forward = 1
* Then execute $ sudo sysctl -p

Add the following rules to iptables

**sudo iptables -t nat -A POSTROUTING --out-interface eth1 -j MASQUERADE**

**sudo iptables -A FORWARD --in-interface eth0 -j ACCEPT**

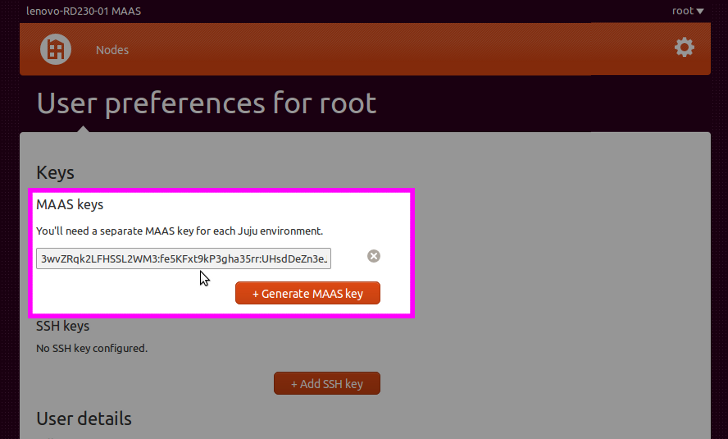
**Note :- you have to do this after every restart or you can do it by script** [**https://help.ubuntu.com/community/Router#Enable\_IP\_forwarding\_and\_Masquerading**](https://help.ubuntu.com/community/Router#Enable_IP_forwarding_and_Masquerading)

All of the forwarded traffic will traverse the FORWARD chain. To filter packets you'll now have to create rules on that chain specifying which interface is incoming/outgoing instead of using the INPUT/OUTPUT chains.

* + 1. now try to wget google.com if looks good then only go ahead.

**JUJU installation : -**

1. To install Juju, you simply need to grab the latest juju-core package from the PPA:
   1. **sudo apt-get install software-properties-common**
   2. **sudo add-apt-repository ppa:juju/stable**
   3. **sudo apt-get update && sudo apt-get install juju-core**
   4. **sudo apt-get install juju-quickstart juju-deployer charm-tools**
   5. **juju generate-config**
2. Obtain Maas API Oauth key for juju



### **Edit the Juju environments.yaml file**

* 1. This is done by generating and editing a file, environments.yaml, which will live in your **~/.juju/** directory. You can generate the environments file manually , but Juju also includes a boilerplate configuration option that will flesh out most of the file for you and minimise the amount of work (and potential errors).
  2. To generate an initial config file, you simply need to run:
  3. juju generate-config
     1. This command will cause a template file to be written to your ~/.juju directory if an environments.yaml file does not already exist. It will also create the ~./juju directory if that does not exist.
     2. If you have previously installed Juju on this machine and the environments.yaml file already exists, you should simply edit it for your new MAAS configuration.
     3. This file will contain sample profiles for different types of cloud services, but you will need only need to edit the section labelled maas, which should look something like this :
     4. **maas:  
         type: maas  
         # Change this to where your MAAS server lives. It must specify the base path.  
         maas-server: 'http://192.168.1.1/MAAS/'  
         maas-oauth: '<add your OAuth credentials from MAAS here>'  
         # default-series: precise  
         authorized-keys-path: ~/.ssh/authorized\_keys # or any file you want.  
         # Or:  
         # authorized-keys: ssh-rsa keymaterialhere**
  4. You may optionally edit the line near the top of the file which reads:
     1. **default: amazon:**
     2. to read:
     3. **default: maas**
  5. The only required information here are the type, maas-server, maas-oauth and one or other of the ways of specifying your ssh keys. The default-series itself defaults to 'trusty' and MAAS will add your SSH keys. So your config should be edited to include the correct address for the MAAS master node and the key which you collected in the previous step:
     1. **maas:  
         type: maas  
         maas-server: 'http://172.16.100.1/MAAS/'  
         maas-oauth: '--MAAS API key string--'  
         authorized-keys-path: ~/.ssh/authorized\_keys**(give directory for .pub) **admin-secret: password** (that will become juju-gui password)

*NOTE:* If you wish to configure your Juju client to additionally work with other cloud environments, please see the [documentation on the main Juju website](about:blank).

### **Check if nodes are in ready state**

add my MAAS default gateway (e.g. 60.60.60.40) to the "Upstream DNS" option in the Settings section of MAAS Web User Interface

## **Juju bootstrap:**

1. before bootstrapping.
2. released all nodes and keep them in ready state using gui.(Appendix 1)
3. run **juju bootstrap -v**
4. it will take more than 20 min if any error doesn’t occur if errors occur get help from Appendix
5. check **juju status**
6. it should show 1 node ‘0’ : status started
7. add another node by **juju add-node**
8. it will take same time as previous.
9. keep checking **juju status**
10. it will show status pending have patience it will become started.
11. finally to are ready to deploy services.

## **Diving in**

1. Once MAAS and Juju are configured you can go ahead and run juju bootstrap. This will provision one of the MAAS nodes and use it as the orchestration node for your juju environment. This can take some time, especially if you don’t have fastpath installer selected, if you get a timeout during your first bootstrap don’t fret! You can increase the bootstrap timeout in the environments.yaml file with the following directive in your maas definition: bootstrap-timeout: 900. During the video I increase this timeout to 900 seconds in the hopes of eliminating this issue.
2. After you’ve bootstrapped it’s time to get deploying! If you care to use the Juju GUI now would be the time to deploy it. You can do so with by running the following command:
   1. **juju deploy --to 0 juju-gui**
3. To avoid having juju spin us up *another* machine we can tell Juju to simply place it on machine 0.

**NOTE**: the --to flag is crazy dangerous. Not all services can be safely co-located with each other. This is tantamount to “hulk smashing” services and will likely break things. Juju GUI is designed to coincide with the bootstrap node so this has been safe. Running this elsewhere will likely result in bad things. You have been warned.

1. Now it’s time to get OpenStack going! Run the following commands:
   1. **juju deploy --to lxc:0 mysql --config openstack.yaml  
      juju deploy --to lxc:0 keystone --config openstack.yaml  
      juju deploy --to lxc:0 nova-cloud-controller --config openstack.yaml  
      juju deploy --to lxc:0 glance --config openstack.yaml  
      juju deploy --to lxc:0 rabbitmq-server --config openstack.yaml  
      juju deploy --to lxc:0 openstack-dashboard --config openstack.yaml  
      juju deploy --to lxc:0 cinder --config openstack.yaml**

To break this down, what you’re doing is deploying the minimum number of components required to support OpenStack, only your deploying them *to* machine 0 (the bootstrap node) in LXC containers. If you don’t know what LXC containers are, they are very light weight Linux containers (virtual machines) that don’t produce a lot of overhead but allow you to safely compartmentalize these services. So, after a few minutes these machines will begin to pop online, but in the meantime we can press on because Juju waits for nothing!

The next step is to deploy the nova-compute node. This is the powerhouse behind OpenStack and is the hypervisor for launching instances. As such, we don’t really want to virtualize it as KVM (or XEN, etc) don’t work well inside of LXC machines.

* 1. **juju deploy nova-compute --config openstack.yaml**

1. That’s it. MAAS will allocate the second, and final node if you only have two, to nova-compute. Now while all these machines are popping up and becoming ready lets create relations. The magic of Juju and what it can do is in creating relations between services. It’s what turns a bunch of scripts into LEGOs for the cloud. You’ll need to run the following commands to create all the relations necessary for the OpenStack components to talk to each other :
   1. **juju add-relation mysql keystone  
      juju add-relation nova-cloud-controller mysql  
      juju add-relation nova-cloud-controller rabbitmq-server  
      juju add-relation nova-cloud-controller glance  
      juju add-relation nova-cloud-controller keystone  
      juju add-relation nova-compute nova-cloud-controller  
      juju add-relation nova-compute mysql  
      juju add-relation nova-compute rabbitmq-server  
      juju add-relation nova-compute glance  
      juju add-relation glance mysql  
      juju add-relation glance keystone  
      juju add-relation glance cinder  
      juju add-relation mysql cinder  
      juju add-relation cinder rabbitmq-server  
      juju add-relation cinder nova-cloud-controller  
      juju add-relation cinder keystone  
      juju add-relation openstack-dashboard keystone**
   2. **juju expose openstack-dashboard**
2. Whew, I know that’s a lot to go through, but OpenStack isn’t a walk in the park. It’s a pretty intricate system with lots of dependencies. The good news is we’re nearly done! No doubt most of the nodes have turned green in the GUI or are marked as “started” in the output of juju status.
3. One of the last things is configuration for the cloud. Since this is all working against Trusty, we have the latest OpenStack being installed. All that’s left is to configure our admin password in keystone so we can log in to the dashboard.
   1. **juju set keystone admin-password="openstack"**

Set the password to whatever you’d like. Once complete, run juju status openstack-dashboard find the public-address for that unit, load it’s address in your browser and navigate to /horizon. (For example, if the public-address was 10.0.1.2 you would go to http://10.0.1.2/horizon). Log in with the username admin and the password as you set it in the command line. You should now be in the horizon dashboard for OpenStack. Click on Admin -> System Panel -> Hypervisors and confirm you have a hypervisor listed.

Step 2:

**Neutron installation:**

<https://jujucharms.com/neutron-gateway/trusty/5>

<https://jujucharms.com/neutron-api/trusty/19>

<https://jujucharms.com/neutron-openvswitch/trusty/2>

above setup uses nova internal network. so nodes unable to ping each other or internet. for that we have to install neutron. note that we have to remove all instances before install neutron.

1. hardware requirement is one more node with 3 nic cards 3 gb RAM, 10 GB HDD will do.
2. to install neutron we have to follow particular sequence as mentioned in above 3 pages. as mentioned nova-compute and nova-cloud-controller should be previously installed. and related.
3. there are 3 major charms that we have to install to turn on neutron. neutron-gateway, neutron-api, neutron-openvswitch
4. Network connection should done as given in hardware diagram.
5. steps
   1. juju deploy neutron-gateway  
      juju add-relation neutron-gateway mysql  
      juju add-relation neutron-gateway**:**amqp rabbitmq-server:amqp  
      juju add-relation neutron-gateway nova-cloud-controller
   2. juju deploy neutron-openvswitch  
      juju add-relation neutron-openvswitch nova-compute  
      juju add-relation neutron-openvswitch neutron-api  
      juju add-relation neutron-openvswitch rabbitmq-server
   3. juju deploy neutron-api --to lxc:0
   4. juju add-relation neutron-api mysql  
      juju add-relation neutron-api rabbitmq-server  
      juju add-relation neutron-api neutron-openvswitch  
      juju add-relation neutron-api nova-cloud-controller
   5. juju set neutron-gateway ext-port=eth1
      1. eth1 will be different if net connetcion is different
   6. if any problem came while creating security group enabled security group by
      1. disable-security-groups=False
      2. juju set neutron-api neutron-security-groups=True

### **Access the console for an Instance**

|  |
| --- |
| **In this video we are going to setup the novnc console to use on our instances.**  **Link to youtube video:** [**https://youtu.be/nY96u2YMpW8**](https://youtu.be/nY96u2YMpW8)  **Good openstack answers doc:**  **<https://ask.openstack.org/en/question/520/vnc-console-in-dashboard-fails-to-connect-ot-server-code-1006/>**  **From the internal JUJU prompt login to the cloud controller:**  **juju ssh nova-cloud-controller/0**  **On the machine that houses the cloud controller install these two packages:**  **sudo apt install nova-novncproxy**  **sudo apt install nova-consoleauth**  **Add this to the nova.conf for the compute nodes via JUJU, using your cloud-controller IP in the URL and the IP on the nova-compute/0 node in the proxyclient:**  **vnc\_enabled=True,novncproxy\_base\_url=http://{nova-cloud-controller-ip}:6080/vnc\_auto.html,vncserver\_listen=0.0.0.0,vncserver\_proxyclient\_address={nova-compute-ip}**  **Add this to the cloud-controller nova.conf file via JUJU, using the cloud controller IP:**  **vncserver\_listen={nova-cloud-controller-ip},vncserver\_proxyclient\_address={nova-cloud-controller-ip},novncproxy\_host=0.0.0.0,novncproxy\_port=6080,novncproxy\_base\_url=http://{nova-cloud-controller-ip}:6080/vnc\_auto.html**  **Make sure the nova.conf got updated correctly, from the internal juju prompt.**  **juju ssh nova-compute/0 sudo cat /etc/nova/nova.conf** |

**Appendix/jugads**

ERROR failed to bootstrap environment: bootstrap instance started but did not change to Deployed state: instance ###### is started but not deployed

1. nano into **.juju/environments.yml** increase **bootstrap-timeout.**

if bootstrap taking too much time on wget.. or failed doing curl.

1. ssh into the node check for /etc/resolve.conf it should have 2 server entries i) maas server ii) external server (8.8.8.8) if it isn’t there add it
2. if permission denied (publickey) after ssh
   1. check if username@ip username is right
   2. Check your \*/etc/ssh/sshd\_config\* file. There, find the line which says
   3. **PasswordAuthentication no**
   4. change it to **yes**.

Re bootstrapping

1. juju destroy-environment maas if not juju destroy-environment maas --force
2. if both doesnot work cd .juju delete environments directory and delete ssh directory.
3. and release node from maas
   1. when node is realised it is not shut down it has to be shut down manually
   2. sudo shutdown -h now
   3. **ssh-keygen -f ".ssh/known\_hosts" -R {node\_ip}** before re acquiring or deploying node and shut down it ..

note : - while deploying always open another terminal and run

$ juju debug-log

to know what’s going on.

**change maas ip from external to internal**

reconfigured the packages :

**$ sudo dpkg-reconfigure maas-region-controller**

**$ sudo dpkg-reconfigure maas-cluster-controller**

**$ sudo reboot**

**In case DNS not working properly**

1. check on maas server /etc/network/interfaces dns-nameservers contains mass server ip.
2. check **/etc/resolv.conf** have entry of maas server in nameservers.

to check lxc containers maid by juju and its ip

**sudo lxc-ls --fancy**

warning does not support kvm / every lxc gatting same ip

ssh to that port

Use the below command to check whether the KVM is installed in the machine or not,

kvm-ok

Output:

INFO: /dev/kvm exists KVM acceleration can be used

Output like the below means,

INFO: Your CPU does not support KVM extensions KVM acceleration can NOT be used

In this scenario turn on virtualization in bios. and check it again.note that for vm over virtualbox in ubuntu nested virtualizer is not yet supported you cannot run lxc on them switch to vmware.

upgrade openstack version chage this.

openstack-origin (string)

cloud:trusty-juno

to login in database

mysql -u root -p`sudo cat /var/lib/mysql/mysql.passwd`

glance db sync error

glance-manage -d -v --log-file= db\_sync

if add-machine throws error

1. no unable to find machine with constraints do maas set constraints to “”
2. if maas error unable to find cluster-controller go to cluster page if it's in error stage  **sudo dpkg-reconfigure maas-cluster-controller to restart**

**Error in launching instance unable to delete:-**

**As pointed out to me by ev0ldave in #openstack the way to fix this is to figure out exactly what the instance ID is.**

**$**mysql -u root -p`sudo cat /var/lib/mysql/mysql.passwd`

**mysql> USE nova;**

**mysql> SELECT id, host, hostname FROM instances;**

**...**

**mysql> UPDATE instances SET deleted=1, deleted\_at="2012-10-10 14:56:30", vm\_state="deleted" WHERE id = 240;**

**...**

**mysql> quit;**

**$ sudo rm -rf /var/lib/nova/instances/instance-00000240**

**Solves the issue assuming your bad instance is ID 240. Rinse and repeat for each instance.**

**It'd be nice to programmatically fix this.**

**turn on rabbitmq management:**

**rabbitmq-plugins enable rabbitmq\_management**

**rabbitmqctl change\_password guest openstack**

***(replace opensatck by password for rabbitmq)***

**turn on new tab on web browser.**

**The web UI is located at: *http://rabbitmq-server-ip:15672/***

**restart nova-compute service check logs /var/log/nova/nova-compute**

**ERROR failed to bootstrap environment: cannot start bootstrap instance: cannot run instances: cannot run instances: gomaasapi: got error back from server: 409 CONFLICT (No available node matches constraints: zone=default)**

**atmecs@ubuntu:~$ server: 409 CONFLICT (No available node matches constraints: zone=default)**

**no ready nodes available to deploy check if at least 1 node in ready state**