MaaS Integration for Baremetal Provisioning in Cloudstack

References

CLOUDSTACK-9913 - Getting issue details... STATUS

Introduction

Metal-as-a-Service (MaaS) is a project by Canonical to enable provisioning of baremetal servers. MaaS has plenty of features which make it very lucrative to be used as a baremetal orchestrator in Cloudstack. Some of the features of MaaS include

- Full API Support
- Official support for CentOS, Ubuntu and Windows images
- IPv6 support
- Can be deployed in a HA configuration
- Network bonding support
- Disk erase support
- Device auto-discovery
- Support for various power types and Chasis
- Disk and partition management

This feature aims to add MaaS as an external baremetal provisioner in Advanced networks in Cloudstack. This **does not** replace the existing baremetal offering by Cloudstack.

High level use cases

1. A user should be able to provision baremetal servers using MaaS in a zone with Advanced networking
2. The provisioned servers should support bonding on the interface (if it is possible)
3. The provisioned servers should be able to be added to multiple networks
4. When the servers are released, the disk should be erased before returning to the user

Document History

Version	Author	Date	Changes
v1.0	Syed Ahmed	09-May-2017	Initial Draft

Functional requirements & non-requirements

1. Allow a configurable option to enable MaaS for baremetal provisioning in Cloudstack
2. Allow a configurable option to pass the information (IP, keys etc) required to connect to Cloudstack
3. Adding a baremetal host by admin in Cloudstack
    a. should also add and commission the host on MaaS
    b. If the host is already present in MaaS it should be picked up by Cloudstack
4. Removing a baremetal host from Cloudstack **should not** remove the host from MaaS. This is done to prevent cases where Cloudstack may run into a bad state, should not affect MaaS

5. When a Baremetal is provisioned with a network, the ToR switches should be configured correctly to reflect the right VLAN
6. When additional networks are
    a. added to the baremetal from Cloudstack, the Top-of-Rack (ToR) switch should also be configured with respective VLANs
    b. removed from the baremetal from Cloudstack, the ToR switches should also release the VLANs
7. A user should be able to add a baremetal template which corresponds to the MaaS image. The URL field in the template specifies which MaaS template to use.
8. A user should be able to delete a baremetal template from Cloudstack
9. When a baremetal is provisioned:
    a. if the host has more than one physical interface connected to the switch as described by the rack config, MaaS will bond the interfaces
    b. if only one physical interface is present, MaaS should use that interface
10. When a baremetal is deleted:
    a. The ToR switch should be configured to remove the VLANs
    b. The disk should be deleted by MaaS.
    c. A config option to select between full disk erase or a fast disk erase may be used

Architecture and Design description

Assumptions

1. MaaS will have its own network where it can PXE boot hosts (PXE network)
2. MaaS will run and control a DHCP server in the PXE network
3. When no VLAN is configured on the ToR switches, the host will be in the PXE network (ie default trunk VLAN is the PXE network)
4. the IPMI is reachable from both Cloudstack and MaaS server
5. MaaS images will be managed independently from Cloudstack
6. CentOS, Ubuntu and Windows are the only OSes that are currently supported
7. Only root disks are supported. No data disk support

Design

Adding MaaS provider

There are two settings to enable MaaS in Cloudstack. The first setting defines which class to use for the baremetal resource

```
external.baremetal.resource.classname = org.apache.cloudstack.compute.maas.MaasResourceProvider
```

This setting will enable MaaS as the handler for baremetal resources instead of <DEFAULT_BAREMETAL_CLASS>

The MaasResourceProvider is configured via the following setting

external.baremetal.system.url = MaasIP=172.31.0.75;MaasKey=fGWswZhpUQrrKUxprD;MaasSecret=CUaKSPXngrtRGRACtLLQKCYLqrQxPgTC;MaasConsumerKey=6sxtWH3XCfKgkspC8J

A typical MaaS key is of the form **<MaasConsumerKey>:<MaasKey>:<MaasSecret>** for example, the above URL is derived from the following MaaS key

6sxtWH3XCfKgkspC8J:fGWswZhpUQrrKUxprD:CUaKSPXngrtRGRACtLLQKCYLqrQxPgTC

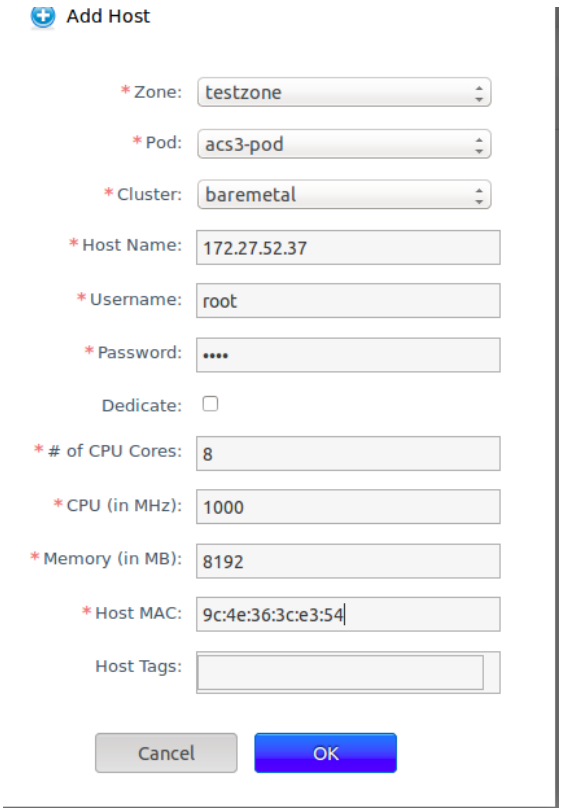
Adding Hosts

Before adding hosts. We will setup the rack configuration. This is a JSON describing the layout of the hosts and how they are connected to the ToR switch. A sample rack config looks something like this:

rack.conf example

```
{
    "racks": [
        {
            "l2Switch": {
                "ip": "172.31.0.109",
                "username": "admin",
                "password": "password",
                "type": "Force10"
            },
            "hosts": [
                {
                    "mac": "00:18:8b:52:79:80",
                    "port": "1/1/1"
                }
            ]
        }
    ]
}
```

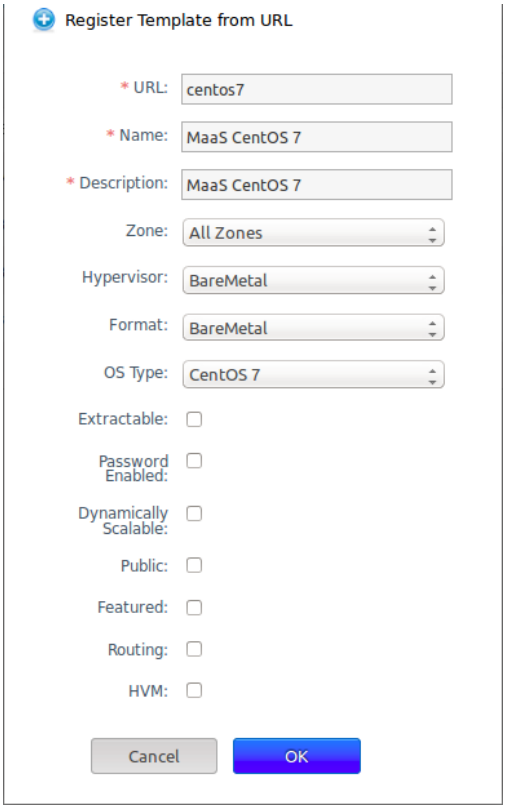
The rack config can be added from the **Global Settings -> Baremetal Rack Configuration** tab. Once this is done, the hosts are added using the **Infrastructure -> Hosts -> Add** button. Make sure that the baremetal cluster is selected (if no baremetal cluster exists, create one first)



The IP/user/password/MAC are the IPMI credentials. CPU, # of Cores and Memory define the dimensions of the physical server. Cloudstack periodically checks the power of the host using IPMI.

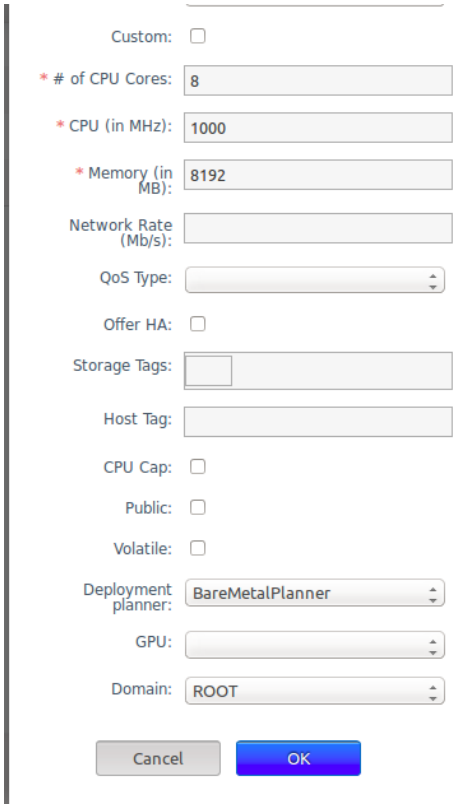
Adding templates

Adding MaaS templates is similar to adding any other template except the **URL** field reflects the name of MaaS image. In the screenshot, **centos7** is the name of the image in MaaS. This Doesn't download the image as it would do in a traditional image but creates a DB entry and the URL is passed to MaaS when the host is provisioned.



Service Offering

When creating a compute offering, make sure that **BareMetalPlanner** is selected and the CPU/Memory details match the host that was added



Create VM Workflow

When a host is provisioned, a **StartCommand** is issued by cloud-server. This command is handled by the **MaasResourceProvider**. The provider verifies a list of constraints like

The host exists in the DB, a default NIC is present on the host, the default NIC belongs to a valid network, the node is already discovered by MaaS. The plugin proceeds to the provisioning stage.There are two possible cases

1. This is a new Baremetal VM that was provisioned
    a. The plugin checks if bonding can be enabled on the host. This is done by looking at the rack config and finding out the interfaces that are connected to the ToR. If there are more than one interfaces connected, the plugin attempts to bond those interfaces together. If it is not possible, then the provisining continues with a single NIC
    b. In both cases, the plugin sets up the MAC address of the interface to the one that is allocated by Cloudstack. This is needed because the DHCP request for this VM can be only recieved if the host's MAC and the mac generated by Cloudstack match.
    c. The next step is to call the **deployMachine** API of MaaS which installs the OS on the baremetal host. Once this call returns, our host is ready to be started
    d. The plugin changes the VLAN on the ToR switch on the interfaces configured to the VLAN of the default NIC. After this, the host is in the correct netowrk.
    e. The plugin sends a command to the IPMI set the first boot device to hard drive (intead of network)
    f. Finally, it sends another IPMI command to boot the host.
    g. Once the host is running,
2. This is an already provisioned Baremetal VM that was in stopped state and is starting now
    a. In this case, the lastHostID field is set on the VM object and it is equal to the hostID of the physical host that handled the StartCommand request and the status of this node in MaaS is also Deployed (instead of Ready)
    b. Since the host is already deployed and in the correct network, the plugin issues an IPMI command to boot the host. When the host boots up, it gets the IP address from the VR that is running in the same network.

Stop VM Workflow

This is very simple. When a user stops a Baremetal VM, cloud-server sends the **StopCommand** to the MaaS plugin. The plugin will just issue an IPMI shutdown command and return back.

Destroy VM Workflow

A new command call **DestroyCommand** is added to Cloudstack. MaaS plugin handles this command and does the following

1. Shuts down the host by issuing an IPMI shutdown command
2. Removes the VLAN from all the interfaces of the host that are connected to the ToR switch. After this, the host is in the PXE network which is accessible from MaaS
3. Change the Boot order via IPMI to boot from network first
4. Issues a **releaseMachine** API call to MaaS which will PXE boot the host and erase the disk

Bonding Support

Bonding support is currently only tested for CentOS. Bonding is only done if there are more than one interface that are connected to the ToR from the physical host. The bonding mode is set to active-backup.

Curtin Setup for Cloud-init

You can modify the default template from MaaS by using preseed scripts. Below is an example of a script that removes cloud-init and sets the root password to "password".

userdata in MAAS (/etc/maas/preseeds/curtin_userdata_centos)

```
root@coe-hq-maas01: /etc/maas/preseeds # cat curtin_userdata_centos
#cloud-config
debconf_selections:
 maas: |
  {{for line in str(curtin_preseed).splitlines()}}
  {{line}}
  {{endfor}}
syed-test:
  - &myscript |
     #/bin/bash

     SCRIPT=$(readlink -f "$0");

     if [ -f "/root/.init_done" ]
     then
             sed -i 's/yum remove -y cloud-init//' /etc/rc.local
             echo "Done done" > /root/.init_message
             exit
     fi
     # yum remove -y cloud-init
     echo "################ REMOVE CLOUD INIT ############ "
     echo "yum remove -y cloud-init"  >> /etc/rc.local
     touch /root/.init_done
     echo "echo "password" | passwd root --stdin" >> /etc/rc.local
 late_commands:
  maas: [wget, '--no-proxy', '{{node_disable_pxe_url}}', '--post-data', '{{node_disable_pxe_data}}', '-O', '/dev
/null']
  test_syed_script: ['curtin', 'in-target', '--', 'sh', '-c', *myscript ]
power_state:
  mode: reboot
```

IP Clearance

No external dependecies are being added for this feature. All code will be developed within Cloudstack’s scope.