ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 3, pp: (158-163), Month: July - September 2015, Available at: www.researchpublish.com

Automated Configuration of Open Stack Instances at Boot Time

N Praveen[1], Dr. M.N.Jayaram[2]

Post Graduate Student[1], Associate Professor[2], EC Department, SJCE, Mysuru, India

Abstract: **Cloud Computing is the buzz word in today's technology era. There are multiple cloud platforms available being open source. Virtualization constitutes the basic ingredient of cloud also automation present along the side. Virtualization and automation goes shoulder to shoulder in enterprise level of a cloud model. OpenStack cloud platform being open source gaining wide popularity providing flexibility for any type of cloud deployments with higher scalability. This paper walks through the cloud-init automation for the OpenStack instances through user-data in the OpenStack cloud platform. Cloud-init being an open source tool provides automation handling the early initialization of cloud instances through user-data script and quickly deploys a cloud server within no time.**

Keywords: **OpenStack Cloud, Instance, Virtual Machine, Cloud-init automation, user-data.**

I. INTRODUCTION

Cloud is a buzzword in today's technology trend. Cloud is a growing idea all over the globe. Cloud runs IT environment and software applications without hosting them at own premises but using a virtual machine with disk space, processing power and bandwidth specification including network functionality like load balancing and firewall setting and much more. Cloud and Virtualization go hand in hand accompanied with automation. Cloud defines the service while virtualization forms the technology at backend. OpenStack is an open cloud operating system for building clouds which began in 2010 as a joint project of RACKSPACE and NASA. It provides organizations an alternative to closed cloud environments, reducing the risk of vendor lock-in associated with proprietary platforms. OpenStack being massively scalable and feature rich, is the strategic choice of many types of organizations. OpenStack Instances are the Virtual Machines hosted from OpenStack deployed system. OpenStack allows launching Virtual Machine instances easily and quickly with in no time. After the instance launch, once the VM is booted the instance can be accessed through VNC by SSH login into the actively running instance server. Then the cloud user must login to the VM, gain access and must do the necessary configurations on to the server based on the requirements or specifications. The configurations may include like the installation of certain packages necessary for the production servers or the development server setup, starting services, or managing the deployment servers. These set of configurations for the instances are done manually after the VM is completely booted and up in traditional way. But the entire process of configuring of an instance can be automated at the boot time of a Virtual Machine easily, which provides a completely configured server as ready package. OpenStack lets cloud users to deploy virtual machines and other instances which handle different tasks for managing a cloud environment on a single fly.

II. RELATED WORK

The OpenStack Instances which are the virtual machines can be termed as cloud based servers can be predominantly behaving as a database server or an application server based on the policies/requirements. Instead of following the regular manual procedure to configure the server system which generally involves the configuration paradigm followed once after the virtual machine is booted and actively up. Cloud-init technology offers early initialization as per the desired needs through the user-data script. There are different ways of OpenStack deployment namely DevStack, PackStack, and

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 3, pp: (158-163), Month: July - September 2015, Available at: www.researchpublish.com

manual deployment procedure and choice selection depends on the business requirements and the underlying platform. This paper focuses on the PackStack deployment mode of OpenStack on RHEL based CentOS7 platform. OpenStack allows users to deploy virtual machines easily. The cloud presents new and interesting things around hardening an instance.

Cloud-init is the industry standard for bootstrapping cloud servers. Cloud-init an Ubuntu package is the defacto multi-distribution package that handles early initialization of a cloud instance. Cloud Images are pre-installed disk images which have been customized by Ubuntu engineering to run on cloud-platforms namely Amazon EC2, OpenStack and so. Cloud-init's behaviour can be configured via user-data. User-data can be given by the user at instance launch time.

This is done via the --user-data or --user-data-file argument during instance launch. Cloud-init, can be considered as a behind the scenes tool that helps configure Cloud images to run in a local hypervisor environment.

III. CLOUD-INIT AUTOMATION

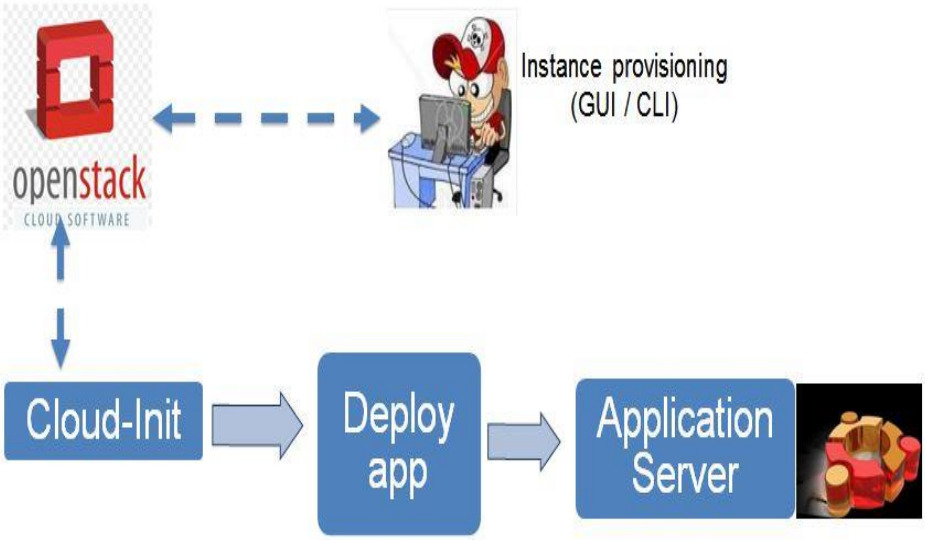

Fig.1: Cloud-init automation

As a prototype, the work presented here concentrates on the cloud-init automation for application deployment as illustrated in Fig1. Application deployment and configuration is automated using user-data script with cloud-init technology. The OpenStack PackStack is successfully deployed at first. Upon this OpenStack cloud, it is possible to provision the virtual machines either using command line interface or easily through the web based GUI dashboard. During the instance launch time, user-data is passed which is used by cloud-init to deploy application server at boot time. As a part of prototype design a simple web application deployment is automated using the cloud-init technology. Here OpenStack server platform used is RHEL based linux and the virtual machine provisioned is from the Ubuntu cloud image. A cloud image is pre-installed disk images which customized by Ubuntu engineering to run on cloud-platforms.

The virtual machine provisioning from OpenStack server can be accomplished either through the GUI horizon dashboard or through the terminal CLI. Approach here is with the terminal using the OpenStack commands. We are provisioning instance or VM server using nova boot command. This command utilizes several arguments information to be passed which must be obtained as a prerequisite from OpenStack deployed system. The syntax with pseudo values are given below to bring conceptual clearity, nova boot --user-data <userdata script> --image <image id/name> --flavor <flavour name/id> --key-name <keypair name> --nic net-id <network_id> Instance_Name

Parameters include:

--user-data <userdata script> for automation (Cloud-init).

--image <image id or name> images are "virtual machine templates." Images can be standard installation media such as ISO images. Essentially, they contain bootable file systems that are used to launch instances (VMs).

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 3, pp: (158-163), Month: July - September 2015, Available at: www.researchpublish.com

--flavor <flavor id or name> A flavor is a resource allocation profile. It specifies how many virtual CPUs and how much RAM the instance will get.

--key-name <keypair name> Keyname of SSH keypair that should be created earlier.

--nic net-id <network id> ID of a network upon which VM server is to be attached.

Instance_Name is the Virtual Machine Name which is user defined name.

Before running the nova boot command for provisioning VM,

Source keystone credentials and authenticating is must

Create the user data script

Know image id/name using command nova image-list

Know flavor id/name using command nova flavor-list

Add the system ssh key nova keypair-add --pub-key ~/.ssh/id_rsa.pub mykey

The id of network onto which the VM will be attached using nova network-list

Also the default security group must be added with security group rules for the application port number. With the key pair, security groups, or rules, instance can be accessed only from inside the cloud through VNC. Even pinging the instance is not possible without these parameters included. The execution of nova boot command provisioning the instance is shown in Figure 2.

```
[root@openstack-aio ~(keystone_admin)]#
[root@openstack-aio ~(keystone_admin)]#
[root@openstack-aio ~(keystone_admin)]# nova boot --user-data /root/cloudinit_script.yaml --image "32c4eb68-d001-4968-b859-9886bba7a4f5" --flavor "m1.small" --k
ey-name mykey --security-groups default --nic net-id="6ff722c6-47ec-479f-ac0c-59e6b1af3227" Praveen_Demo2
+--------------------------------------+--------------------------------------------------------------+
| Property                             | Value                                                        |
+--------------------------------------+--------------------------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                                                       |
| OS-EXT-AZ:availability_zone          | nova                                                         |
| OS-EXT-SRV-ATTR:host                 | -                                                            |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | -                                                            |
| OS-EXT-SRV-ATTR:instance_name        | instance-0000001f                                            |
| OS-EXT-STS:power_state               | 0                                                            |
| OS-EXT-STS:task_state                | scheduling                                                   |
| OS-EXT-STS:vm_state                  | building                                                     |
| OS-SRV-USG:launched_at               | -                                                            |
| OS-SRV-USG:terminated_at             | -                                                            |
| accessIPv4                           |                                                              |
| accessIPv6                           |                                                              |
| adminPass                            | UHvUvDhVJv44                                                 |
| config_drive                         |                                                              |
| created                              | 2015-06-09T07:54:21Z                                         |
| flavor                               | m1.small (2)                                                 |
| hostId                               |                                                              |
| id                                   | 18bb8516-5680-416d-acd3-c0c5ae149780                         |
| image                                | UbuntuTrusty Server (32c4eb68-d001-4968-b859-9886bba7a4f5)   |
| key_name                             | mykey                                                        |
| metadata                             | {}                                                           |
| name                                 | Praveen_Demo2                                                |
| os-extended-volumes:volumes_attached | []                                                           |
| progress                             | 0                                                            |
| security_groups                      | default                                                      |
| status                               | BUILD                                                        |
| tenant_id                            | fbfc245e859b4610a4db12b7f49a4b44                             |
| updated                              | 2015-06-09T07:54:21Z                                         |
| user_id                              | be32180a0e5b4ae29be8ede9d87ea28d                             |
+--------------------------------------+--------------------------------------------------------------+
[root@openstack-aio ~(keystone_admin)]#
```

Fig.2 Instance Provisioning by command

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 3, pp: (158-163), Month: July - September 2015, Available at: www.researchpublish.com

Depending on the command parameters provided in the nova boot CLI, the command returns a list of server properties. A status of BUILD indicates that the instance has started, but is not yet online. A status of ACTIVE indicates that the instance is active. After the floating/public IP assignment for the virtual machine servers, the servers are ready for accessing from external. Instance can be observed through logging in to the horizon OpenStack dashboard as in Figure 3. Figure 3 shows the list of instances (VM servers) assigned with floating IP for external access. With this public IP, one can access the VM server running web app which is deployed at VM's boot time.

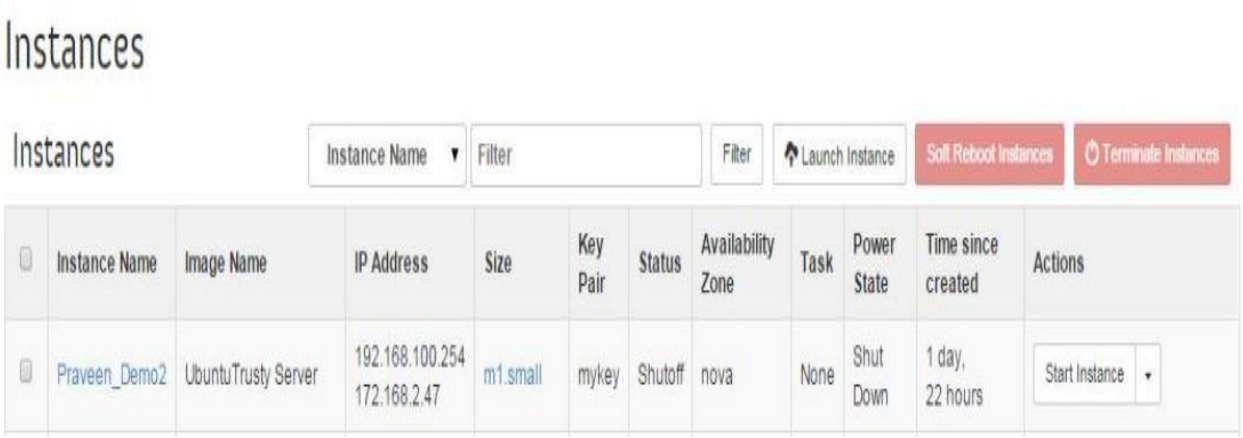

Fig.3 Instance List

Cloud-init script implementation can be done in multiple ways but the popular one cloud-config is followed during this prototype work. Cloud-init script being a YAML script follows strict indentation and developed with modular approach. The script follows declarative syntax.

Module 1: used the chpasswd attribute to set/change the server login credentials as per the requirements. During the instance boot up time, the control passes through the mentioned login credentials under the chpasswd attribute once after the instance is up.

Module 2: used the write_files attribute to possess the root permission and modify by appending the system files. The necessary contents to modify the system file needs root user permission. The root user being the owner is permitted to both read & write, allowing all other users only to read the file which is specified present along the path parameter

Module 3: used the apt_update attribute which is being set to true to ensure that all system packages are all new updated ones on the virtual machine.

Module 4: This part of the script uses package attribute and lists the packages to be installed like python-pip which is a python package manager, git package.

Module 5: The previous modules provide background setup for the application deployment. Module 5 constitutes the application deployment works. Here the attribute named runcmd is used which is responsible for the execution of below procedural steps:

1. Navigating to the desired directory
2. Cloning the source repo which pulls the actual source files related to the application to be deployed.
3. Installing the requirements, the necessary package dependencies for the application running.
4. Enable the application to run at desired port number of the server and run the application.

Also it is important aspect that the necessary changes or modifications needed to be done with the required permissions on desired files and package dependencies installation purely depends on the web app chosen for deployment and upon the business requirements or policies.

Once after the provision of virtual machine server the cloud-init script executes and successfully completes. This can be found either means of dashboard or through terminal. When accessed through the dashboard, the active log outputs are as shown in Figure 4.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 3, pp: (158-163), Month: July - September 2015, Available at: www.researchpublish.com

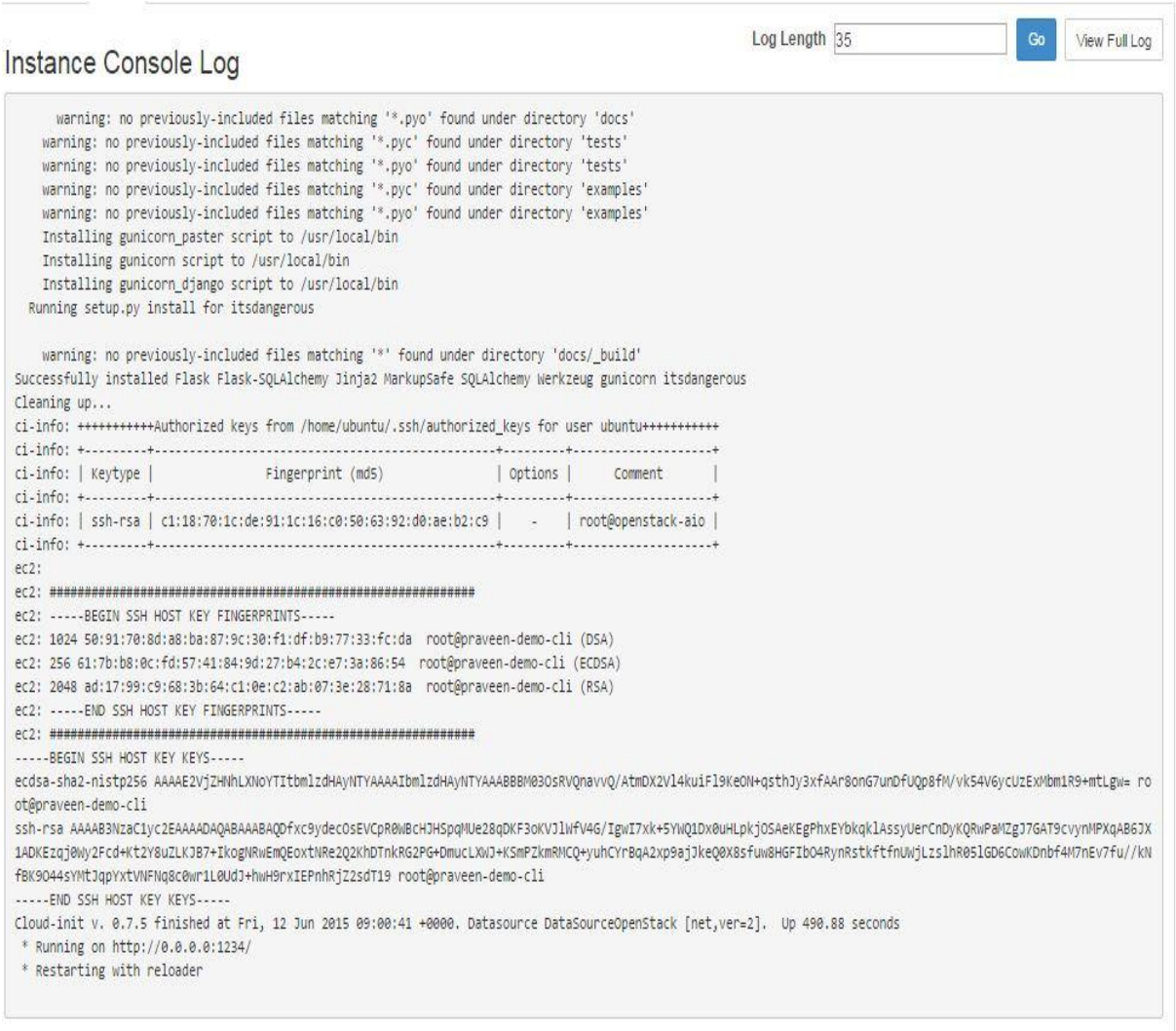

Fig .4 Active Log Output for cloud-init

IV. ADVANTAGES OF CLOUD-INIT AUTOMATION

1. Creation of the VM and the tasks done when it is up can all be scripted and automated eases work of sysadmin.

2. With Cloud-init, it is possible to pass the executable actions to instances at launch time through the user-data fields and can configure and customize the instances at launch time.

3. Able to interact with different providers like Amazon or the OpenStack metadata service.

4. Responsiveness to business needs: customers have dramatically reduced the time it takes them to deliver applications into production when run cloud-init in conjunction with another provisioning system like Ansible, Chef, Puppet or Salt.

5. Cloud Service automation reduces project costs for both cloud providers and users.

6. Supports easy maintenance of cloud servers eliminating the repetition of multiple rework and increases productivity.

V. LIMITATIONS OF CLOUD-INIT AUTOMATION

1. If cloud-in it fails because of some errors in the script file or doesn't contain all of the needed directives, a new instance must be created and launched again

2. The Cloud-init script is passed by using user data field has a size limit of 16K bytes and challenging to audit since it is script based.

VI. OVERCOMING THE LIMITATIONS

Cloud-init with automation functionality makes the life easier for sysadmin. But testing and debugging the script can be quite intensive if you need to boot up a machine every time. Restarting the failed instance with a new cloud-init file will not work. To handle and overcome the limitations, there are easy approaches available. In the user data field, the cloud-init script can be modularly implemented integrating with other automation tools overcoming the code memory limitation size. Also the YAML script which is user-data for cloud-init once developed can be verified by using freely available YAML validator before implementing.

VII. CONCLUSION

Customizing Instances with Cloud-init and user data makes the cloud user or sysadmin work easier. During launching of an instance user data is passed to customize it at instance launch time with cloud-init technology. The user data is passed as a base64 encoded string and is available locally on the instance when the instance is running. Because applications are installed and configured at launch time, the applications may take longer to get up and running. In some cases, such as scaling out an application with Auto Scaling, the additional time to have an instance in service might be delayed because they are launched when you need to meet additional demand. To speed up the instance launch, a hybrid solution can be used where some of the core, stable application components are built into the image (such as base infrastructure services) and components that change more frequently are installed at launch time. This provides the flexibility to easily upgrade the application version on new virtual machine servers provisioned. By combining cloud deployments with Cloud-init, there exists the full control of the trade-offs between instance launch time and flexibility to change application components.

REFERENCES

[1] An Oracle White Paper, “Oracle Cloud Computing”, May 2010

[2] Official Cloud-init documentation, https://cloudinit.readthedocs.org/

[3] OpenStack Foundation, “OpenStack Cloud Administrator Guide”, January, 2015

[4] OpenStack Foundation, “OpenStack Virtual Machine Image Guide”, January, 2015

[5] Cloud-init documentation from the Ubuntu community, https://help.ubuntu.com/community/CloudInit

[6] Red Hat Customer Portal, Frequently Asked Questions about cloud-init,https://access.redhat.com/articles/rhel-atomic-cloud-init-faq.