Table 1: **Introduction to HPC on Openstack**

1 Introduction

Term HPC as a Service refers to an on demand instantiation of HPC Service in a Cloud. This guide presents a simple HPC cluster instantiation procedures in an existing OpenStack based System. HPC as a service relies on two main principals to instantiate HPC service 1. Providing pre-build OS images for compute nodes with HPC optimized software and 2. Uses of Cloud-Init to configure and tune HCP services. This recipe provides a simple guides to build HPC optimized OS images, prepare cloud-init recipes and finally instantiate fully functional HPC System using HPC optimized image and cloud-init. For HPC System recipe instantiate HPC master node (aka sms node) and HPC compute nodes using pre-configured OS images. The terms master and SMS are used interchangeably in this guide OS Images are build using components from OpenHPC software stack. OpenHPC represents an aggregation of a number of common ingredients required to deploy and manage an HPC Linux* cluster including resource management, I/O clients, development tools, and a variety of scientific libraries. These packages have been pre-built with HPC integration in mind using a mix of open-source components. The documentation herein is intended to be reasonably generic, but uses the underlying motivation of a small, 4-node statefull cluster installation to define a step-by-step process. Several optional customizations are included and the intent is that these collective instructions can be modified as needed for local site customizations. Base Linux Edition: this edition of the guide highlights installation without the use of a companion configuration management system and directly uses distro-provided package management tools for component selection. The steps that follow also highlight specific changes to system configuration files that are required as part of the cluster install process.

TargetAudience

This guide is targeted at experienced Linux system administrators for HPC environments. Knowledge of software package management, system networking, PXE booting and OpenStack system software is assumed. Command-line input examples are highlighted throughout this guide via the following syntax:

[sms] `# echo "OpenHPC hello world"`

Unless specified otherwise, the examples presented are executed with elevated (root) privileges. The examples also presume use of the BASH login shell, though the equivalent commands in other shells can be substituted. In addition to specific command-line instructions called out in this guide, an alternate convention is used to highlight potentially useful tips or optional

Table 2: **Bare Metal Preparation**

This guide uses diskimage-builder utility to build and configure OS images for sms node as well compute node. Preparing images is an optional part of overall recipe. If user has predefined images then environment variable chpc_create_new_image must be reset and path to images must be provided using environment variable chpc_image_deploy_kernel, chpc_image_deploy_ramdisk, chpc_image_user, and chpc_image_sms. In this example cloud images are build on controller node [ctrlr]# " Once images are build, they are stored at standard openHPC Path.
[ctrlr]# `CHPC_CLOUD_IMAGE_PATH=/opt/ohpc/admin/images/cloud/`

2.1 Install and Setup disimage-builder

Images can be built on any supported OS.
In this example we will install and build images on controller node, user can do same on a system independent of their production cluster.
Install diskimage-builder and its dependencies from base OS distro
[ctrlr]# `yum y install diskimage-builder PyYAML`
Install grub dependency
[ctrlr]# `yum y install parted`
Diskimage-builder installed from base distro does not have group install feature. So add a patch (Note: this probably will be included into RPM and will be part of that rpm installation)
[ctrlr]# `yum y install <DIB patch>`
Setup common environment for diskimage-builder
diskimage-builder, or dib, uses environment variables and elements to customize the images. For debugging purpose, we will create default user `chpc` with a password `intel8086`, with sudo privilege. These variables are used by element devuser.
[ctrlr]# `export DIB_DEV_USER_USERNAME=chpc` [ctrlr]# `export DIB_DEV_USER_PASSWORD=intel8086` [ctrlr]# `export DIB_DEV_USER_PWDLESS_SUDO=1`
Now add path to custom elements which are not part of base diskimage-builder. OpenHPC provides few HPC elements. [Note: This also can be part of openHPC provided rpm for dib. In that case remove this step]
[ctrlr]# `export ELEMENTS_PATH="$(realpath ../../dib/hpc/elements)"`
Add path to HPC specific files [note: same as earlier, this too can be part of rpm package]
[ctrlr]# `export DIB_HPC_FILE_PATH="$(realpath ../../dib/hpc/hpc-files/)"`
HPC elements are common for OpenHPC and Intel HPC Orchestrator, environment variable DIB_HPC_BASE tell dib which one to pick. For OpenHPC set environment variable
[ctrlr]# `export DIB_HPC_BASE=ohpc`
Make sure openhpc package is installed. ohpc-slurm is one of the input setup

Table 3: **Cloud Init Preparation**

[12pt]article graphicx

4 CloudInitPrep

OpenStack uses cloud-init for boot time initialization of cloud instances. This recipe relies on cloud-init to initialize HPC instances in an OpenStack cloud. This recipe prepares cloud-init initialization template script, which is than updated with sms-ip and other environment variables just before the provisioning. This is than fed as user data to Nova during instance creation. Script generated here will be executed by root during boot.
Preparing template for compute node cloud-init
Create an empty chpc_init file and open for editing. You can also use existing template and modify. Start editing by adding some environment variable, first one is to set path to shared folder for cloud-init

```
chpcInitPath=/opt/ohpc/admin/cloud_hpc_init

logger "chpcInit:  Updating Compute Node with HPC
configuration"
```

Update rsyslog configuration file to send all the syslog to sms. Sms_ip is the tag used here is updated with IP of SMS node just before provisioning.

```
# Update rsyslog

cat /etc/rsyslog.conf | grep "<sms_ip>:514"

rsyslog_set=$?

if [ "$rsyslog_set" -ne "0" ]; then

echo "*.* @<sms_ip>:514" >> /etc/rsyslog.conf

fi

systemctl restart rsyslog

logger "chpcInit:  rsyslog configuration complete,
updating remaining HPC configuration"
```

Assuming sms node nfs share /home, /opt/ohpc/pub, =/opt/ohpc/admin/cloud_hpc_init lets mount them during boot

```
# nfs mount directory from SMS head node to Compute Node

cat /etc/fstab | grep "<sms_ip>:/home"
```

Table 4: **Instantiation**

[12pt]article graphicx

5 Instantiating

To instantiate OpenHPC system, we will first prepare openstack components with HPC images, networking and other relevant configurations. After the configuration we will instantiate HPC head node and HPC compute node using nova. It is assumed that system admin has installed OpenStack controller services and OpenStack network services (i.e. keystone, nova, ironic, glance, neutron, mongodb, rabbitmq server, heat etc). Controller node is configured with Openvswith Bridge on internal network port. Two tenant name admin and services are created in keystone to manage the services. All the services are created by system admin. Below is expected endpoint list.

```
openstack service list
```

+———————————-+———+————+————+ — ID — Region — Service Name — Service Type — +— ———————————-+———+————+————+ — d5aeeb54713745c29ed3c2e4a97f59bd — RegionOne — ironic — baremetal — — 86c71badbf8b4446a1b699eef05f3f41 — RegionOne — nova — compute — — 70d138db26214d0bbc6b3ade8bf6f6f8 — RegionOne — gnocchi — metric — — f34c3a58b9c648aaacabeeefd589a0d2 — RegionOne — neutron — network — — 789c3fb6f9ae4e249ee4023484ccb5fc — RegionOne — aodh — alarming — — 2531392e2d084b4582b364572e79a7b5 — RegionOne — heat — orchestration — — c183b73f654e454eaf5784c4b98149d8 — RegionOne — Image Service — image — — 850b3c2943df4dca99338ff2013f657b — RegionOne — cinder — volume — — 81cefa79212a4780abe5a1da281a0172 — RegionOne — novav3 — computev3 — — 36a6a7c7968a4a94bea07c8e30fa5c4b — RegionOne — keystone — identity — — db70a8676dd44dd09b3ada7475e67383 — RegionOne — cinderv3 — volumev3 — — c4161d4c9c6b4080b2cb66c2f580853d — RegionOne — ceilometer — metering — — d5714e8adb094671ad0388d04214c44d — RegionOne — cinderv2 — volumev2 — +———————————-+———+————+————+

```
openstack project list
```

+———————————-+———-+ — ID — Name — +———————————-+———-+ — 7464fcc8f1b34048bd09fe165d18647b — admin — — b1ed7efb53cc44c8b06daaee15b6a296 — services — +———————————-+———-+

Recipe below is tested with controller node installed and configured using packstack. Reference section provide more detail on packstack installation