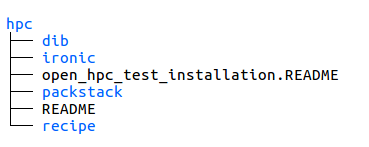
# About

This document provides top level overview of the implementation of Cloud-HPC proof of concept. It mainly covers two use cases explained below.

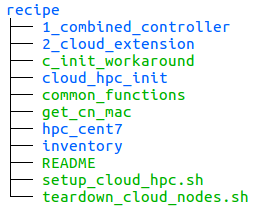
# Package directory structure



Disk image builder elements for creating HPC compute node images

Scripts and inventory files for cloud – POC use case 1 and 2

**Figure 2‑1 Top level directory structure of cloud POC implementation**



Scripts for use case 1: HPC in Cloud

Scripts for use case 2: Cloud extension

Cloud init template script that needs to be customized for a HPC head node

Common functions used by the top level scripts

Utility script used to obtain MAC addresses of a node using its BMC IP

HPC orchestration software (OpenHPC or Intel HPC Orchestrator) installation scripts customized to exclude Warewulf provisioning

Cloud inventory files: Defines the cloud controller node specific details

Master script which the administrator invokes to setup usecase 1 or usecase 2

Performs clean up of Openstack services created

**Figure 2‑2 Recipe directory structure**

# Master script common path for use case 1 and use case 2

No

No

No

Yes

Yes

Yes

./setup\_cloud\_hpc.sh <args>

hpc\_cent7/intel/recipe.sh

Install HPC Orchestrator on Controller/Head Node

hpc\_cent7/ohpc/recipe.sh

Install HPC Orchestrator on Controller/Head Node

Validate and source the input file of HPC software’s recipe.sh

Validate and source the cloud inventory file

Install OpenHPC?

common\_functions: setup\_comptename()

Populate: c\_name[], cc\_name[]

Change the hostname of controller/head node to one specified in cloud inventory

Install IntelOrch?

../packstack/recipe/packstack-install.sh

Install OpenStack components

Install OpenStack?

common\_functions: setup\_hosts(), setup the hostname resolution for sms/head node, cloud compute nodes

**Figure 3‑1 Master script (setup\_cloud\_hpc.sh) control flow – Part I**

No

No

Yes

Yes

1\_combined\_controller/set\_os\_hpc

Use case 1?

Use case 2?

2\_cloud\_extension/set\_os\_hpc

sinfo

Run hostname -i on all compute nodes using srun

**Figure 3‑2 Master script (setup\_cloud\_hpc.sh) control flow – Part II**

# Use case 1: HPC in cloud

The scripts corresponding to this use case are placed in hpc/recipe/1\_combined\_controller. After handling the common path the master script branches to set\_os\_hpc script specific to a usecase. In this case control branches to hpc/recipe/1\_combined\_controller/set\_os\_hpc.

## set\_os\_hpc

prepare\_cloud\_init

prepare\_chpc\_image

return

start

prepare\_chpc\_openstack

deploy\_chpc\_openstack

update\_cnodes\_to\_sms

**Figure 4‑1 set\_os\_hpc for usecase 1: hpc/recipe/1\_combined\_controller**

## prepare\_cloud\_init

Create a temporary directory on Controller node/HPC head node.

return

start

Copy the cloud init template script (hpc/recipe/cloud\_hpc\_init/<orch or ohpc>/chpc\_init) to the temporary directory created in the previous step

a

Customize cloud init script (chpc\_init) for the given HPC head node:

* Replace the template IP address with the HPC head node’s IP

Update slurm configurations (slurm.conf) on the HPC head node:

* Add cloud node descriptions
* Add a partition with cloud nodes

Copy following files on the HPC head node to temporary directory create in first step:

/etc/passwd,

/etc/shadow,

/etc/group,

/etc/slurm/slurm.conf,

/etc/pam.d/slurm,

/etc/munge/munge.key,

/etc/hosts

NFS share temporary cloud init directory created in first step on Controller/HPC head node

**Figure 4‑2 Control flow for 1\_combined\_controller/prepare\_cloud\_init**

## prepare\_chpc\_image

prepare\_user\_image

return

start

prepare\_deploy\_image

Prepare HPC compute node image by adding custom elements to disk-image-builder command line

Prepare a CentOS 7 diploy image

**Figure 4‑3 Control flow for 1\_combined\_controller/** **prepare\_chpc\_image**

## prepare\_chpc\_openstack

This script installs and configures ironic for baremetal provisioning on the local CentOS 7 machine using the OpenStack-Mitaka release. It assumes that OpenStack-packstack has already been installed on the system and the keystonerc\_admin file is in the user’s $HOME directory. It is assumed that the user running the script is root or running with sudo permissions. The script itself creates the baremetal\_admin and baremetal\_observer roles, installs and configures tftp, sets up PXE boot and restarts neutron, nova, and ironic.

## deploy\_chpc\_openstack

This script sets up the environment necessary to boot compute nodes and boots the compute nodes. The script assumes naming for networks, subnets, user images, etc. The script itself checks if the following are present, and creates them if they are not: network labeled “sharednet1”, subnet labeled “subnet01”, images labeled “user-image”, “deploy-vmlinuz”, and “deploy-initrd”, nova flavor labeled “baremetal-flavor”, ironic nodes labeled (node prefix)[1-(number of nodes)], ironic ports associated for each of the ironic nodes, nova keypair labeled ostack\_key. The script also updates the quota for the admin user and boots the nodes using nova.

## update\_cnodes\_to\_sms

No

No

Yes

Yes

Add cloud nodes to /etc/genders on Controller/HPC head node

return

start

Enable mrsh services (mshell, mlogin) on Controller/HPC head node

Enable mrsh?

Enable cluster shell?

Update cluster shell config with cloud nodes on Controller/HPC head node

Update slurm configuration file’s ‘ControlMachine’ entry with the hostname of Controller/HPC head node

Update slurm configuration file’s of Controller/HPC head node with the cloud nodes and create a partition – normal with these cloud nodes

Restart slurmctld deamon on Controller/HPC head node

Set state of all compute node to ‘idle’ using scontrol

**Figure 4‑4 Control flow for 1\_combined\_controller/** **update\_cnodes\_to\_sms**

# Use case 2: Cloud extension

The scripts corresponding to this use case are placed in hpc/recipe/ 2\_cloud\_extension. After handling the common path the master script branches to set\_os\_hpc script specific to a usecase. In this case control branches to hpc/recipe/2\_cloud\_extension/set\_os\_hpc.

## set\_os\_hpc

prepare\_cloud\_init

prepare\_chpc\_image

return

start

extend\_hpc\_to\_openstack

update\_cnodes\_to\_sms

**Figure 5‑1 set\_os\_hpc for usecase 1: hpc/recipe/2\_cloud\_extension**

## prepare\_cloud\_init

return

start

Update slurm configurations (slurm.conf) on the HPC head node:

* Add cloud node descriptions

Copy following files on the HPC head node to cloud init directory created on the HPC head node:

/etc/passwd,

/etc/shadow,

/etc/group,

/etc/slurm/slurm.conf,

/etc/pam.d/slurm,

/etc/munge/munge.key,

/etc/hosts

NFS share temporary cloud init directory created in first step on Controller/HPC head node

Copy updated slurm.conf files to all compute nodes of HPC cluster

Restart slurmctld

Create and prepare Cloud init directory HPC head node

Customize cloud init script (chpc\_init) for the given HPC head node:

* Replace the template IP address with the HPC head node’s IP

**Figure 5‑2 Control flow for 2\_cloud\_extension/prepare\_cloud\_init**

## prepare\_chpc\_image

This script is functionally same as the prepare\_chpc\_image explained in section 4.3

## extend\_hpc\_to\_openstack

The purpose of this script is to copy the required files from an HPC cluster to a cloud controller node that are required to provision a cloud node with HPC capabilities. These files include an HPC-specific cloud-init script, an HPC image the cloud controller can use to boot, any keys necessary to ssh into the cloud nodes. Once those files are copied over, the script invokes deploy\_chpc\_openstack (Section 4.5) through a remote ssh call, which then sets up the environment and boots the cloud compute nodes on the cloud controller.

## update\_cnodes\_to\_sms

No

No

Yes

Yes

Add cloud nodes to /etc/genders on Controller/HPC head node

return

start

Enable mrsh services (mshell, mlogin) on Controller/HPC head node

Enable mrsh?

Enable cluster shell?

Update cluster shell config with cloud nodes on Controller/HPC head node

Find the existing slurm partition for HPC cluster’s compute nodes from /etc/slurm.conf

Set state of all compute node (HPC Cluster + Cloud Provisioned nodes) to ‘idle’ using scontrol

Extend the HPC cluster’s slurm partition with cloud provisioned nodes: Append the cloud node names to HPC node names and dynamically update the HPC cluster partition using scontrol with concatenated node string

**Figure 5‑3 Control flow for 2\_cloud\_extension/update\_cnodes\_to\_sms**

# Cloud-init

Cloud-init is a protocol used to initialize the cloud provisioned nodes with a specific user data. At the final stage of cloud node provisioning the cloud bare-metal nodes are booted using nova. As part of the nova boot, the cloud init script that was customized (during prepare\_cloud\_init stage) for the given HPC head node is passed as user data --user-data). This injects the cloud init script into the cloud provisioned node and as part of the boot up every cloud node will execute this script. Following flow chart explains the functionality of the cloud init script when executing on the cloud node.

Yes

No

Yes

No

No

Yes

start

Update /etc/rsyslog.conf with the HPC head nodes’s IP address, restart rsyslog

Enable rsyslog?

Is NFS mount of /home set

Update /etc/fstab for NFS mounting /home from HPC Head node

Is NFS mount of /opt/<>/pub set?

Update /etc/fstab for NFS mounting /home from HPC Head node

NFS mount,

/home

/opt/<ohpc or orch>/pub

Is NFS mount cloud init set?

Update /etc/fstab for NFS mounting cloud init directory from HPC Head node

NFS mount,

cloud init directory

**Figure 6‑1 Control flow of cloud init script on a cloud provisioned compute node during the boot up – Part I**

return

Restart NFS

Enable NTPD, set HPC head node as NTP server and restart NTPD

Copy following files from the cloud init directory of HPC head node to their corresponding location at /etc on cloud compute node

<NFS mounted cloud init>/passwd 🡪 /etc/passwd,

<NFS mounted cloud init>/shadow 🡪 /etc/shadow,

<NFS mounted cloud init>/group 🡪 /etc/group,

<NFS mounted cloud init>/slurm.conf 🡪 /etc/ slurm/slurm.conf,

<NFS mounted cloud init>/slurm 🡪 /etc/pam.d/slurm

<NFS mounted cloud init>munge.key 🡪 /etc/munge/munge.key

<NFS mounted cloud init>/hosts 🡪 /etc/hosts

Enable munge

Restart munge

Enable slurmd

Restart slurmd

Setup the hostname as configured on HPC head node

Enable and restart munge, slurmd

Set read-write permission on /etc/ssh/ssh\_host\_\*\_key

**Figure 6‑2 Control flow of cloud init script on a cloud provisioned compute node during the boot up – Part II**