Rapid Deployment of Bare-Metal and In-Container HPC Clusters Using OpenHPC playbooks

Joshua Higgins, Taha Al-Jody and Violeta Holmes
HPC Research Group
University of Huddersfield, UK

HPC Systems Professionals Workshop (HPCSYSPROS18)

Outline

- Motivation
- OpenHPC
- Ansible
- Clusterworks
- Ansible Playbooks for Clusterworks
- Cluster Deployment (bare metal)
- In-container Cluster Deployment
- Summary

Motivation

- HPC is expected to encompass a wide range of applications
- Software environments of the resources should be flexible and easily re-configurable
- Configuration management is used by administrators of machines at many scales
- However, few provide practical solutions that are easily accessible to the wider community
- Hence, Cluster Works - a toolbox of Ansible roles and playbooks to easily deploy cluster software stacks

Configuration Management

- Process of defining a systems
  - physical,
  - functional and
  - operational attributes
- Existing tools for building HPC systems:
  - Puppet
  - Chief
  - Ansible
  - SaltStack

OpenHPC

- Provides a full stack of HPC software components for cluster architecture
- Aids administrators in deploying combination of
  - Compilers,
  - MPI libraries,
  - User interface and
  - Environment modules
- Procedures for building clusters from scratch

Ansible

- Ansible is an open source IT configuration management, deployment, and orchestration tool
- It is distinctive from other management tools in many respects, aiming to provide large productivity gains to a wide variety of automation challenges
- Ansible performs automation and orchestration of IT environments via Playbooks

Clusterworks

- Toolbox of Ansible roles and playbooks
- Used to deploy cluster software stack
- OpenHPC recipes used for validated packages for the software stack
- Workflows for provisioning HPC cluster software environments
- Installation of a Beowulf-style cluster

Playbooks and YAML

- **Playbooks** are a **YAML** definition of automation tasks that describe how a particular piece of automation should be done
- **Ansible Playbooks** are prescriptive, responsive descriptions of how to perform an operation
- In case of IT automation it clearly states what each individual component of IT infrastructure needs to do
- **YAML** (YAML Ain't Markup Language) is a human-readable data serialization language
- It is commonly used for configuration files

Ansible Playbooks

- Ansible Playbooks consist of series of '**plays**' that define automation across a set of hosts, known as the '**inventory**'
- Each 'play' consists of multiple '**tasks**,' that can target one, many, or all of the hosts in the inventory
- Each task is a call to an Ansible module - a small piece of code for doing a specific task
- These tasks can be simple, such as placing a configuration file on a target machine, or installing a software package
- They can be complex, such as spinning up an entire CloudFormation infrastructure in Amazon EC2

Ansible Playbooks for Clusterworks

- As part of the Clusterworks toolbox, Ansible playbooks were created to include well defined tasks and roles
- The roles are grouped in high-level tasks:
  - Master/head node installation
  - Slave/worker node installation
  - Updating nodes post-installation
- Global *config* file allows parameters to be set to determine the components installed in the environment

Clusterworks Ansible Roles

- Ansible roles to deploy
  - Stateful or stateless cluster using the xCAT provisioning middleware, and
  - PBS Professional as the resource management middleware.
- Implemented to support CentOS
- It supports configuring xCAT as part of the cluster installation
- Automatically configure the required definitions in the xCAT database for the nodes to be installed, based on the options chosen in the configuration file

Install master	Install nodes	Update nodes
validation	validation	validation
repos	xcat_ mkdef	repos
ohpc_base	pbs_create	ntp
xcat_base		sync_files
nfs		nfs
pbs		ssh
ssh		ohpc base
dev_tools		pbs

- Extreme Cluster Administration Toolkit
- Open source
- Scales up to 100,000 nodes
- Automates installation of cluster nodes
- Services for machine discovery, network identification and remote installation
- xCAT can be used to deploy machines in
  - stateful (installed to a local hard disk) or
  - stateless mode, where provisioning occurs over PXE

xCAT operation

- Suite of Command Line Instructions (CLI)
- Central database with
  - definitions of each node,
  - configuration profiles,
  - network settings and
  - OS images
  - For example

    lsdef -t node

    lists each node registered in the xCAT database
- Operations over many objects at once

Secure Shell (SSH)

Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network

- The standard TCP port for SSH is 22
- The best known example application is for remote login to computer systems by users
- SSH uses public-key cryptography to authenticate the remote computer and allow it to authenticate the user, if necessary

Cluster security

- Passwords are supported, but SSH keys with ssh-agent are one of the best ways to use Ansible
- Root logins are not required, you can login as any user, and then *su* or *sudo* to any user
- Ansible's "authorized_key" module is a way to use Ansible to control what machines can access what hosts

Steps using the playbooks:

1) With a working Python installation, install Ansible using

   pip install ansible

2) Clone the **clusterworks/inception repository** from GitHub

3) Copy the *config* template and adjust to suit your environment,

   configuring the SMS/head node network identification and path to the CentOS image

4) Edit the inventory to include details of the Master/head and worker nodes

5) Run the playbook *install_master*

6) Run the playbook *install_nodes*

7) Boot and install the worker nodes via the network

8) Run the playbook *update_nodes*

Cluster installation completion

- When all steps are complete, the cluster will be ready.
- ***pbsnodes*** command can be used to inspect the cluster status from the head node.
- Users could now be created
- Users can submit jobs for execution on a cluster

In-container Cluster Deployment

- The same roles can be reused within Ansible Container in order to generate a Docker image, rather than installing on a physical cluster
- Possible to quickly and easily package a known working configuration within a container
- Portable and flexible way to create, test and share software stacks

- Playbook for Ansible Container
  - Builds a container which includes
    - OpenHPC repositories,
    - base packages, and
    - development tools
- Same roles used to install run-time applications on the physical cluster can be used to install in the container.

```
version: "2"
settings:
  conductor:
    base: centos:7
  project_name: ohpcdemo
services:
  demo:
    from: centos:7
    roles:
      - repos
      - ohpc_base_compute
      - dev_tools
    entrypoint: /bin/bash
registries: {}
```

Summary

Clusterworks toolbox key features:

- Built on the work by the OpenHPC Community
- Easy to use workflows for provisioning and deploying cluster environments
- Repository, package and configuration management
- Turn-key, extensible and instilled with best practice
- Containerize an environment to share or deploy in the cloud
- 100% free and open-source software

- The **inception** repository https://github.com/clusterworks/inception provides the core Ansible playbook
- It is used to build a cluster environment using a well-defined, easy to use and extensible workflow
- To deploy on bare-metal, just provide an inventory of the physical resources
- To deploy in the cloud, a container can be created from environment configuration using Ansible Container.

Thank you

- Questions?