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
Huddersfield, UK
{joshua.higgins,taha.al-jody,v.holmes}@hud.ac.uk

***Abstract*—In this paper, we present a toolbox of reusable Ansible roles and playbooks in order to configure a cluster software environment described by the freely available OpenHPC recipes. They can be composed in order to deploy a robust and reliable cluster environment, instilled with the best practise offered by the OpenHPC packaging, and the repeatability and integrity guarantees of the configuration managed approach. With container virtualization setting a new trend in scientific software management, we focus this effort on supporting the deployment of such environments on both bare-metal and container-based targets.**

***Index Terms*—hpc, middleware, ansible, containers**

I. Introduction

Nowadays, HPC is expected to encompass a wide range of applications, in addition to the standard batch scheduling execution paradigm. It is essential that the software environment of the resource is flexible and easily re-configurable, but in a way which is controlled and reproducible. Configuration management is a proven technology used by administrators of machines at many scales, from on-premise hardware to cloud resources, in order to manage the setup and maintenance of the software environment.

Traditionally, the configuration of HPC systems has either grown organically, based around the needs of a specific user community, or configured using vendor-provided tools. Configuration management approaches are becoming popular, but are often opinionated, and it is a daunting task in order to encapsulate the cluster building process from scratch. There is a large amount of literature that considers the experience of institutions adopting DevOps-like approaches for cluster management, but few that provide practical solutions that are easily accessible to the wider community with a low technical burden.

In this paper, we introduce *The Cluster Works* - a toolbox of Ansible roles and playbooks that can be used to deploy cluster software stacks. The OpenHPC recipes are used as a foundation to provide dependable and validated packages for the software stack. The goal of the project is to create easy to use workflows for provisioning and deploying HPC cluster software environments, using a single well-defined methodology that supports applying the configuration to both ”bare-metal” (whether physical or virtual) and container-based targets. We detail the reference installation on a Beowulf-style cluster at the University of Huddersfield, provide a guided example on how to use the playbooks to deploy your own cluster, and finally, invite the community to participate in our open source project on GitHub.

II. Background

A. Configuration Management

Configuration Management is an engineering process in order to define a system in terms of the functional, physical and operational attributes that exist throughout its lifetime. In the context of software, this allows the definition and scale out of systems quickly and reliably, providing methods to quantify the effectiveness of the configuration changes - for example, a bad change can be rolled back to a well known configuration in order to restore the affected service [1].

A large number of open source and commercial configuration management tools are available that can potentially be useful in the HPC context. In [2], a method for evaluating a configuration management system is proposed, including an assessment of tools available at the time. They identified that there is no association between the maturity of a tool and the feature set that is provides, encouraging adoption of a tool that meets the requirements regardless of longevity. This approach can be a challenge in the HPC context, as software environments are traditionally glacial in the rate of change. However, it also identifies limitations in the existing tools which may be significant in our community: very few tools support high level abstractions, such as *”create 3 worker nodes”*, provide limited integration with version control systems, and do not allow access controls in order to delegate domains of administration to different users.

In the era of cloud computing and DevOps, four tools stand out as having both large user communities and a wide installation base: Puppet [3], Chef [4], SaltStack [5] and Ansible [6]. These tools have previously been considered as a solution for building HPC systems [7]–[10]. However, the existing approaches are often not easily reproducible by others, as they are an exemplar of each implementation, rather than a generalized framework of reusable building blocks. In addition, many solutions consider automating the deployment of HPC infrastructure in clouds [11]–[13] but are often not directly transferable to on-premise or other types of systems that do not provide the required APIs for discovering resources and configuration.

B. OpenHPC

OpenHPC [14], [15] provides a full stack of HPC software components for a reference cluster architecture. It includes packages for provisioning and resource management middleware, compilers, parallel libraries, popular scientific software and performance benchmarking tools. The project aim is to encourage community participation and exchange of best practice HPC ingredients and knowledge, supported by key academic and commercial organizations in the industry.

Using OpenHPC, a system administrator can deploy validated combinations of compilers, MPI libraries and supporting software with familiar user interfaces, such as environment modules. In addition, the project also publishes recipes for building the reference cluster platform from scratch, which documents the caveats and recommended procedures that may not be obvious to a seasoned system administrator without significant experience in HPC and scientific computing.

C. xCAT

Extreme Cluster Administration Toolkit (xCAT) is an open source deployment and provisioning middleware, which has been demonstrated to scale up to 100,000 nodes [16]. It automates the installation of cluster nodes, providing services for machine discovery, network identification and remote installation, to enable efficient and direct management of the cluster at scale. xCAT can be used to deploy machines in stateful - installed to a local hard disk - or stateless mode, where provisioning occurs over PXE. Several Linux distributions, Windows and Hypervisor Operating Systems are supported for deployment, allowing the administrator to create and maintain images for machines with range range of roles within the HPC estate.

xCAT is operated through a suite of CLI tools and a central database which holds the definitions of each node, configuration profiles, network settings and OS images. For example, `lsdef -t node` lists each node registered in the xCAT database. Each tool supports operations over many objects at once, and can be easily scripted by the administrator following the Unix philosophy.

D. Container virtualization

Container virtualization has quickly become an indispensable tool in HPC for turning software environments into a portable package that can be executed on a system regardless of the underlying software stack. Using tools such as Singularity [17], Charliecloud [18] and Docker [19], the community is busy publishing containers that encapsulate the run time environment of many scientific codes. One of the disadvantages is the lack of discipline in building the container environment, choosing the base OS for a container image, and to what extent you include core system dependencies. For example, some containers must be executed on an already existing HPC cluster, whilst some can run on any machine that simply provide a container run time. Despite this potential inconsistency, containers provide scope for deploying highly flexible HPC systems, and significantly reduce the barrier of entry for the average user to virtualization technology.

III. Why Ansible for HPC?

In this section we outline the motivation for adopting Ansible as the preferred configuration management tool for an HPC system.

Firstly, Ansible respects the already established security principles within the cluster, as it uses SSH to perform configuration tasks on remote systems. It does not require to establish a second Public Key Infrastructure just for the purpose of configuration management. Therefore, it is possible to delegate administration tasks simply by authorizing the user's key on the appropriate machine and using *sudo* to provide fine grained control over elevated privileges, and thus, which configuration actions can be managed by a particular user.

Secondly, it does not require an agent to be installed on every node. This means that it does not generate a constant stream of messages over the network as machines check that the configuration is compliant, and the technical burden in order to install Ansible within a cluster is very low. In HPC, these are especially critical factors, as we do not want to compromise the execution performance, nor require significant effort to install agents on a potentially vast estate of machines.

Furthermore, the configuration is defined in YAML files stored on the file system. They are plain text and easily stored in version control, rather than requiring a dedicated server component and databases. Therefore, administration tasks can be performed from any machine with a copy of the files and which can reach the target nodes - rather than designating a central provider, which can be inconvenient in geographically distributed systems such as a grid.

The configuration is abstracted into roles, which are composed of tasks. Roles are grouped together into playbooks. The targets of a playbook are defined at run time by supplying an inventory of hosts. Tasks can be parameterized to maintain a high level of abstraction within the configuration, adding host-specific information to the inventory which is substituted at run time. The administrator is free to define the granularity of each role or playbook - for example, a worker node role could contain every task to configure a worker node. However, separate roles could also be defined, such as *NFS client, LDAP client, PBS client* and composed together within a worker node playbook. In future, the same roles could be reused, such as for a login/UI node playbook.

The declarative approach of the configuration allows the administrator to define the desired state in terms of *package X should be present and installed*, and Ansible will invoke the distribution package manager with the appropriate command line. This avoids having many `if..else` statements to support different platforms that would be required with a script-based approach.

Finally, Ansible is unique in that it can configure container images as the target instead of physical hosts, without any additional supporting infrastructure, and push the resulting image to the Docker Hub. This provides a mechanism which can automatically package an application and HPC software stack already defined in a playbook into a container image without additional complexity.

IV. Ansible playbooks for OpenHPC recipies

The Ansible playbooks have been created by following the OpenHPC version 1.3.5 [20] recipes and breaking down each task into logical roles. The roles are grouped together in high level tasks that cover master / head node installation, node installation and updating nodes post-installation. A global config file allows parameters to be set which determine the components that are installed in the environment.

The implementation of each role follows a consistent pattern, where distribution specific tasks are separated into different files and conditionally imported into the main file. This enables the playbook to be easily ported in future to support other flavours of Linux.

Each playbook begins with validation tasks that ensure the environment is prepared appropriately before installation. This highlights problems such as the hostname not set in */etc/hosts*, a firewall blocking required ports, or that the IP address defined for the head node is actually assigned to a valid interface on the system. In addition, it performs sanity checking of the requested components, warning the user if no compiler has been selected, or if a default environment module has not been defined. When deploying an environment on a many different systems, this provides some visibility of the reproducibility challenges that might arise, and avoids frustration from a configuration that fails part way through.

A. Available roles

The initial version of the playbooks support a subset of the available OpenHPC recipes, providing Ansible roles to deploy a stateful or stateless cluster using the xCAT provisioning middleware, and PBS Professional as the resource management middleware. The playbook incorporates support for configuring xCAT as part of the cluster installation. Therefore, it is will automatically configure the required definitions in the xCAT database to allow the nodes to be installed, based on the options chosen in the configuration file. The tasks have initially been implemented to support RedHat / CentOS derived distributions only, as shown in Table I.

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

TABLE I
Available roles

B. Example deployment

In this section, we outline the steps necessary to deploy a standalone cluster using the playbooks on a bare-metal system running CentOS 7.x.

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 head and worker nodes
5) Run the `install_master` playbook
6) Run the `install_nodes` playbook
7) Boot and install the worker nodes via the network
8) Run the `update_nodes` playbook

After each step is complete, the cluster will be ready and the `pbsnodes` command can be used to inspect the cluster status from the head node. A user could now be created and begin to submit jobs for execution.

As this procedure uses the OpenHPC recipe without any customization, the standalone cluster will share home directories from the head node using NFS. The local password database is copied to every node when the `update_nodes` playbook is executed. A limitation of this methodology is that updates will only be applied to nodes which are powered on and accessible. However, the idempotent nature of the playbooks means they can be executed as often as required in order to maintain the desired state of the cluster. Therefore, this problem can be alleviated by executing the `update_nodes` playbook periodically or in response to node boot events from xCAT.

V. Considerations for production usage

At the University of Huddersfield, we deployed a 32 node Beowulf cluster in under 30 minutes using the Ansible playbook in the default configuration. If an inventory of the nodes, MAC addresses and IP addresses is already available, the cluster deployment is automated from start to finish. For a temporary cluster used for the purpose of a single experiment, or for demonstration or training purposes, the default recipe is likely sufficient, and a good exercise in order to gain experience with how to compose a HPC software stack without putting a production system at risk. However, it will be necessary to incorporate customizations to the environment so that it can be used in production or at scale.

For example, in a production environment, the shared storage may not be hosted directly on the head node. In addition, a directory server such as LDAP may be used to provide single sign on capabilities for the cluster. Configuration of these aspects is out of the scope of the OpenHPC recipe, and thus has not been implemented within the playbook.

```
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: {}
```

Fig. 1. Container Playbook Example

We expect that the base OpenHPC Ansible roles will be combined with a set of custom roles made by the administrator, to provide contextually specific configuration such network file systems, directory servers or sysctl tuning parameters. Custom roles can be included within the `install_master` and `update_nodes` playbooks to facilitate this customization. The file structure provides separation between base roles and custom roles, which will allow upgrades and changes to be tested independently, and it is essential to store the configuration within a version control repository. Furthermore, this provides a mechanism for administrators to test and contribute custom roles to the base set for use by the community.

VI. Deploying In-Container HPC Stack with Ansible Container

The playbook provides roles that configure the OpenHPC repositories and installs a set of development tools, libraries and software to support HPC applications. The same roles can be reused within Ansible Container [21] in order to generate a Docker image, rather than installing on a physical cluster. Therefore, it is possible to quickly and easily package a known working configuration within a container, without fragmenting the configuration into two separate components to handle bare metal and container installation respectively. This offers a portable and flexible way to create, test and share software stacks without placing an additional burden on the administrator.

An example playbook for Ansible Container is shown in Figure 1. This will build a container which includes the OpenHPC repositories, base packages, and the development tools selected in the playbook configuration - compilers, libraries and default environment. The same roles used to install run time applications on the physical cluster can be used to install in the container. When the container is deployed on another system, the user can be confident that the application will execute as intended.

This opens up several possibilities; firstly, whilst the container format itself is standardized and well portable between systems, the process of building the container image - and what should be included within an image - is not. The playbooks define recipes for building containers using the same established methods used to deploy and manage software on physical systems, addressing this gap. Secondly, the container offers a robust mechanism in order to achieve Continuous Integration (CI) and Continuous Deployment (CD) of the HPC software stack itself. The container environment, as it can be configured in exactly the same way as the physical system, can be used to test and validate that the cluster setup described by the playbook is functional and properly integrated - rather than testing individual components in isolation. Finally, the container can be exploited to easily create virtual instances of existing compute nodes. This can be applied in order to extend the physical cluster into a dynamically scalable resource, such as a cloud, to facilitate bursting. Therefore, the combination of Ansible Container and the OpenHPC playbook is a natural progression, reducing the barrier of entry to take advantage of DevOps techniques and capabilities within the HPC environment.

VII. Summary

In this paper, a suite of Ansible roles and playbooks is presented that can be used in order to build and deploy a cluster environments using a well defined, easy to use and extensible workflow based on components maintained by the OpenHPC Community. The rationale for selecting Ansible as the configuration management tool for a HPC system is outlined, showing that it offers a low barrier to adoption by respecting already established principles such as SSH authentication and plain text configuration files.

An example deployment, including the process to reproduce the cluster installation on another system, is documented. This allows a turn-key batch scheduling cluster to be deployed in an automated manner, instilled with the best practice packaging of the OpenHPC repositories.

Finally, we described how the same roles and playbooks can be reused to generate container images for HPC applications, with controlled and consistent behaviour. It allows them to be packaged along with the supporting middleware and libraries required for execution, enabling enhanced reproducibility, portability, automated testing and flexible deployment from bare metal to the cloud.

The playbooks are released under an open source license and available on GitHub [22].

VIII. Future Work

The available roles are currently being developed to include the additional recipes offered by the OpenHPC community, to include the full range of provisioning and resource management middlewares, extra packages and Linux distribution support. CI and CD pipelines will be established for the playbooks in order to rapidly deploy, test and work towards hardening the solution for production.

Based on the properties of xCAT, which is proven to scale up to 10,000+ cores, and the fact that Ansible uses an SSH agentless configuration approach, we expect that the OpenHPC playbooks will have good scalability from small, to medium and extreme scale systems. It is planned to evaluate the performance on a large, production HPC cluster.

Finally, 3 use cases for Ansible Container using the OpenHPC playbooks were presented, which are currently in use by the authors. This will be further developed to ensure that the functionality is on par with the bare metal deployment method, offering the capability to deploy the same turn-key cluster stack within a virtualized environment.

REFERENCES

[1] C. C. Group. (2017, jun) The importance of configuration management. [Online]. Available: https://c2sconsultinggroup.com/the-importance-of-configuration-management/

[2] T. Delaet, W. Joosen, and B. Vanbrabant, "A survey of system configuration tools," in *Proceedings of the 23rd Large Installations Systems Administration (LISA) conference*. Usenix association, 2010, pp. 1–14.

[3] Puppet: deliver better software faster. [Online]. Available: https://puppet.com/

[4] Chef: Configuration management. [Online]. Available: https://www.chef.io/configuration-management/

[5] Saltstack: intelligent, event-driven it automation software. [Online]. Available: https://www.saltstack.com/

[6] Ansible: Simple it automation. [Online]. Available: https://www.ansible.com/

[7] V. Hendrix, D. Benjamin, and Y. Yao, "Scientific cluster deployment and recovery–using puppet to simplify cluster management," in *Journal of Physics: Conference Series*, vol. 396, no. 4. IOP Publishing, 2012, p. 042027.

[8] S. Varrette, P. Bouvry, H. Cartiaux, and F. Georgatos, "Management of an academic hpc cluster: The ul experience," in *Proc. of the 2014 Intl. Conf. on High Performance Computing Simulation (HPCS 2014)*. IEEE, 2014.

[9] J. Slawinski and V. Sunderam, "Autonomic multi-target deployment of science and engineering hpc applications," in *Cloud and Autonomic Computing (ICCAC), 2014 International Conference on*. IEEE, 2014, pp. 180–186.

[10] D. Franois, "Behind the scenes of a foss-powered hpc cluster at uclouvain," available at https://archive.fosdem.org/2018/schedule/event/hpc_uclouvain/.

[11] C. Campbell, N. Mecca, I. Obeid, and J. Picone, "The neuronix hpc cluster: Cluster management using free and open source software tools," in *Signal Processing in Medicine and Biology Symposium (SPMB), 2017 IEEE*. IEEE, 2017, pp. 1–3.

[12] G. D. S. Aguiar, A. E. M. Brito, F. Fonseca, L. E. T. Silva, and L. L. Vieira, "Cloudslurm: a multi-provider approach for hpc in the cloud," 2018.

[13] W. C. Proctor, M. Packard, A. Jamthe, R. Cardone, and J. Stubbs, "Virtualizing the stampede2 supercomputer with applications to hpc in the cloud," *arXiv preprint arXiv:1807.04616*, 2018.

[14] (2018, sep) Openhpc. [Online]. Available: https://openhpc.community/

[15] K. W. Schulz, C. R. Baird, D. Brayford, Y. Georgiou, G. M. Kurtzer, D. Simmel, T. Sterling, N. Sundararajan, and E. Van Hensbergen, "Cluster computing with openhpc," 2016.

[16] (2018, sep) xcat. [Online]. Available: https://xcat.org/

[17] G. M. Kurtzer, V. Sochat, and M. W. Bauer, "Singularity: Scientific containers for mobility of compute," *PloS one*, vol. 12, no. 5, p. e0177459, 2017.

[18] R. Priedhorsky and T. Randles, "Charliecloud: Unprivileged containers for user-defined software stacks in hpc," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2017, p. 36.

[19] D. M. Jacobsen and R. S. Canon, "Contain this, unleashing docker for hpc," *Proceedings of the Cray User Group*, 2015.

[20] (2018, sep) openhpc/ohpc 1.3.5ga - github. [Online]. Available: https://github.com/openhpc/ohpc/tree/v1.3.5.GA

[21] (2018, feb) Ansible container documentation. [Online]. Available: https://docs.ansible.com/ansible-container/

[22] (2018, aug) clusterworks/inception github repository. [Online]. Available: https://github.com/clusterworks/inception

APPENDIX

A. Abstract

In this paper, we present a toolbox of reusable Ansible roles and playbooks in order to configure a cluster software environment described by the freely available OpenHPC recipes. They can be composed in order to deploy a robust and reliable cluster environment, instilled with the best practise offered by the OpenHPC packaging, and the repeatability and integrity guarantees of the configuration managed approach. With container virtualization setting a new trend in scientific software management, we focus this effort on supporting the deployment of such environments on both bare-metal and container-based targets.

B. Description

1) Check-list (artifact meta information):

- **Program: Ansible, xCAT, PBS Professional, CentOS 7**
- **Run-time environment: Python, Shell**
- **Hardware: 3 Physical or Virtual Machines**
- **Publicly available?: Yes**

2) How software can be obtained: Open source software via GitHub (https://github.com/clusterworks/inception), also tarball included in artifact.

3) Hardware dependencies: Physical or Virtual Machines.

4) Software dependencies: CentOS 7, Python, Ansible

C. Installation

See Section IV B of the paper.

D. Evaluation and expected result

Cluster is deployed using the playbook based on the provided inventory and configuration, which can be used to compile MPI codes and submit jobs for execution.

E. Experiment customization

The repository contains a default configuration that deploys xCAT and PBS. The user can supply additional Ansible roles that can apply customizations to the deployed environments on the head and worker nodes. Depending on the environment, it may be necessary to customize the proxy, repository and package settings before installation of the playbook. These customizations are documented in the configuration file.

F. Notes

The video artifact provides a guided walkthrough in order to reproduce the installation, evaluation and expected result.