International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Deployment and Installation of the NFS Server using Ansible Automation Configuration Management Tool

Melad Mohamed Salim Elfighi[1], Younis Wrodko Ahmed Mohammed[2], Osama Abdalla Elmahdi Elghali[3]

[1, 2, 3]College of Technical Science - Sebha

Abstract: ***Configuration management tools are essential for maintaining consistency, reliability, and efficiency in IT infrastructure. Among these tools, Ansible stands out due to its simplicity, powerful automation capabilities, and agentless architecture. Ansible uses human-readable YAML templates to define configuration states, enabling both system administrators and developers to manage and deploy applications efficiently. This abstract explores the core features of Ansible, including its playbooks, inventory management, and modularity through roles. Additionally, it highlights the benefits of using Ansible in diverse environments, such as reduced deployment times, minimized configuration drift, and enhanced collaboration between teams. By leveraging Ansible's robust and scalable architecture, organizations can achieve seamless infrastructure as code (IaC) practices, ensuring consistent and repeatable system configurations. Aim of this paper is to utilize Ansible for the automated deployment and installation of the NFS Server (NFS) on remote servers.***

Keywords: Ansible, Cloud, Automation, NFS, SSH

1. Introduction

Configuration Management in Industry was a purely manual task that is to be done by System Administrator. Automation helps in replacing repetitive tasks and helps in saving time, money, and increasing productivity. But the Industry is now changing a lot with the popularity of DevOps, Cloud Computing, and new Automation Tools. DevOps is in demand nowadays as it shortens the life cycle of Software Development. With today's demand for automation, consistency, and the move towards cloud and DevOps, companies from different sectors are adopting easy-to-use tools that help them to achieve their goal by reducing complexities. Therefore, it is of the utmost importance for a company to have its services installed, configured, and running as quickly as possible and as consistent as possible to help reduce costs.

In Today's world the organizations want to shift from manual work to automation to decrease the cost of releasing software. To compete in the market, organizations want to release their software earlier and with better frequency. DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). Main Goal of DevOps is to reduce the time between development and operation of software without affecting the quality. IBM has coined the term Collaborative DevOps as" designing processes for coordinating software development teams with IT operations teams". There are various examples of organization practicing Devops including Flickr, Netflix and Etsy. All spawn and Hammond. By adopting DevOps and variants thereof, many organizations have improved the software delivery which increases the productivity. It consists of various stages such as continuous development, continuous integration, continuous testing, continuous deployment, and continuous monitoring. Teams that adopt DevOps culture, practices, and tools become high-performing, building better products faster for greater customer satisfaction.

In the rapidly evolving landscape of IT infrastructure, the need for efficient, consistent, and scalable management solutions is paramount. Ansible, an open-source automation tool, has emerged as a leader in configuration management, application deployment, and task automation. Developed by Michael DeHaan and first released in 2012, Ansible has gained widespread popularity due to its simplicity, flexibility, and powerful capabilities.

Ansible distinguishes itself with its agentless architecture, which eliminates the need for additional software on managed nodes. Instead, it utilizes standard SSH connections and Python scripts to execute tasks, making it easier to set up and maintain compared to traditional agent-based systems. This approach not only reduces overhead but also enhances security and reliability by leveraging existing infrastructure components.

At the core of Ansible's functionality are playbooks, which are written in YAML, a human-readable data serialization standard. Playbooks define a series of tasks that describe the desired state of a system, enabling users to automate complex processes with minimal effort. This declarative approach ensures idempotency, meaning that running the same playbook multiple times will always yield the same result, thus preventing configuration drift and ensuring consistency across environments.

Ansible's modular design further enhances its versatility. Users can extend its functionality through custom modules, and the community-driven Ansible Galaxy provides a rich repository of reusable roles and playbooks. This modularity allows Ansible to integrate seamlessly with a wide range of technologies and platforms, from cloud services and containers to networking devices and bare-metal servers.

In addition to its technical strengths, Ansible fosters a collaborative and inclusive community. The open-source nature of the project encourages contributions from users worldwide, leading to continuous improvement and

innovation. The extensive documentation and active community forums provide valuable resources for both beginners and experienced users, facilitating knowledge sharing and best practices.

This introduction explores the fundamental principles and components of Ansible, highlighting its impact on modern IT operations. By automating repetitive tasks, reducing errors, and promoting infrastructure as code (IaC) practices, Ansible empowers organizations to achieve greater efficiency, agility, and scalability in their IT operations. As businesses continue to embrace digital transformation, Ansible's role as a cornerstone of infrastructure automation and management becomes increasingly significant.

Ansible employs a human-readable language, YAML, for defining configuration states, enabling users to manage and deploy applications seamlessly across diverse environments. Unlike traditional configuration management tools that rely on agent-based architecture, Ansible operates over standard SSH connections, simplifying the deployment process and reducing the overhead associated with maintaining additional software on managed nodes.

This research paper delves into the core features of Ansible, examining its architecture, functionality, and practical applications. By exploring the use of playbooks, roles, and inventory management, this paper aims to illustrate how Ansible facilitates Infrastructure as Code (IaC) practices, ensuring consistency, repeatability, and scalability in managing IT infrastructure. Furthermore, it investigates the advantages of Ansible in various deployment scenarios, emphasizing its role in enhancing operational efficiency, reducing configuration drift, and fostering collaboration across teams.

As organizations continue to seek agile and reliable solutions for managing their growing IT ecosystems, Ansible's relevance and impact become increasingly significant. This paper aims to provide a comprehensive understanding of Ansible's capabilities, showcasing its potential to transform infrastructure management through automation and streamlined operations.

2. Objective & Aim of the research

The objective this research paper is to utilize Ansible for the automated deployment and installation of the NFS Server (NFS) on remote servers. This involves creating an Ansible playbook that will:

1) Ensure that the target remote servers are reachable and properly configured for Ansible automation.
2) Update the package manager repositories to guarantee the latest version of NFS is installed.
3) Install the NFS package (nfs-utils-1.3.0-0.68.el7.x86_64) on the remote servers.
4) Configure the NFS service to start on boot and ensure that it is running.
5) Deploy a basic configuration file for NFS to demonstrate its operational state.
6) Verify the successful installation and operation of the NFS service on all targeted remote servers.

By achieving this objective, the process will demonstrate the efficiency and reliability of Ansible in managing software deployments across multiple remote systems, ensuring consistency and reducing manual intervention.

3. Key features of Ansible

Agentless Architecture: Ansible does not require any agents or additional software to be installed on the managed nodes. It uses SSH for communication, which simplifies the setup and maintenance process.

Idempotency: Ensures that operations are applied only when necessary, avoiding redundant actions and maintaining the desired state.

Modules: Ansible uses modules to accomplish specific tasks. Modules can manage system resources, install software, or interact with APIs. Users can also write custom modules.

Playbooks: Playbooks are YAML files that define a series of tasks to be executed on remote hosts. They are the heart of Ansible's configuration, deployment, and orchestration.

Roles: Roles are a way to organize playbooks and related files. They facilitate reuse and sharing of configurations.

Inventory: An inventory file lists the hosts and groups of hosts that Ansible manages. It can be static or dynamic

4. Architecture & Component of Ansible

The Ansible automation engine consists of various components as described below as follows.

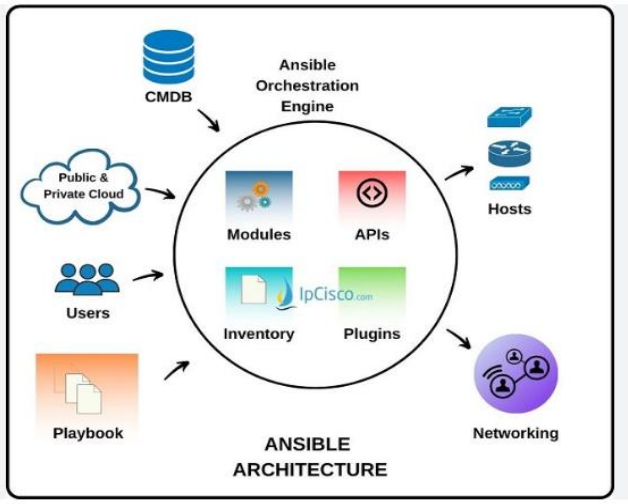


Figure 1: Architecture of Ansible tool

Inventories:
Ansible inventories are lists of hosts with their IP addresses, servers, and databases which have to be managed via an SSH for UNIX, Linux, or Networking devices, and WinRM for Windows systems.

APIs:
Application Programming Interface or APIs are used as a mode of transport for public and private cloud services.

Modules:
Modules are executed directly on remote hosts through playbooks and can control resources like services, packages, files, or execute system commands. They act on system files, install packages and make API calls to the service network. There are over 450 Ansible that provide modules that automate various jobs in an environment. For example, Cloud Modules like Cloud Formation create or delete an AWS cloud formation stack.

Plugins:
Plugins are pieces of code that augment Ansible's core functionality and allow executing Ansible tasks as a job build step. Ansible ships with several handy plugins and one can also write it on their own. For example, Action plugins act as front-ends to modules and can execute tasks on the controller before calling the modules themselves.

Networking:
Ansible uses a simple, powerful, and agent-less automation framework to automate network tasks. It uses a separate data model and spans different network hardware.

Hosts:
Hosts refer to the nodes or systems (Linux, Windows, etc) which are automated by Ansible.

Playbooks:
Playbooks are simple files written in YAML format which describe the tasks to be executed by Ansible. Playbooks can declare configurations, orchestrate the steps of any manual ordered process and can also launch various tasks.

CMDB:
It stands for Configuration Management Database (CMDB). In this, it holds data to a collection of IT assets, and it is a repository or data warehouse where we will store this kind of data, and It also defines the relationships between such assets.

Cloud:
It is a network of remote servers hosted on the internet to store, manage and process data instead of storing it on a local server.

Component of Ansible:
Ansible automates the management of remote systems and controls their desired state.

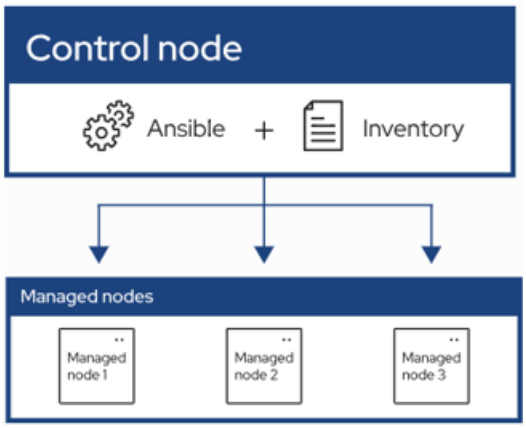


Figure 2: Comprehensive component of Ansible

As shown in the preceding figure, most Ansible environments have three main components:

Control node:
A system on which Ansible is installed. You run Ansible commands such as ansible or ansible-inventory on a control node.

Inventory:
A list of managed nodes that are logically organized. You create an inventory on the control node to describe host deployments to Ansible.

Managed node:
A remote system, or host, that Ansible controls.

Use Cases

- **Configuration Management:** Ensuring that systems are configured to a desired state.
- **Application Deployment:** Automating the deployment of applications across multiple servers.
- **Task Automation:** Automating repetitive IT tasks such as backups, system updates, and user management.
- **Orchestration:** Coordinating multiple systems and services to work together in complex deployments.
- Benefits
- **Ease of Use:** Simple syntax and agentless architecture make it easy to learn and implement.
- Scalability: Suitable for managing both small and large-scale environments.
- **Consistency:** Ensures that configurations are applied uniformly across all managed nodes.
- Flexibility: Can manage a wide variety of systems and applications, both on-premises and in the cloud.
- **Community Support:** Large, active community contributing to modules, roles, and best practices.

5. Simulation setup and installation of NFS server on worker node using Ansible configuration automation tools

In this paper, we are using VMware Workstation 16 Pro tool for setup server, VMware Workstation Pro is a hosted (Type 2) hypervisor that runs on x64 versions of Windows and Linux operating systems for networking we are using bridge Network.

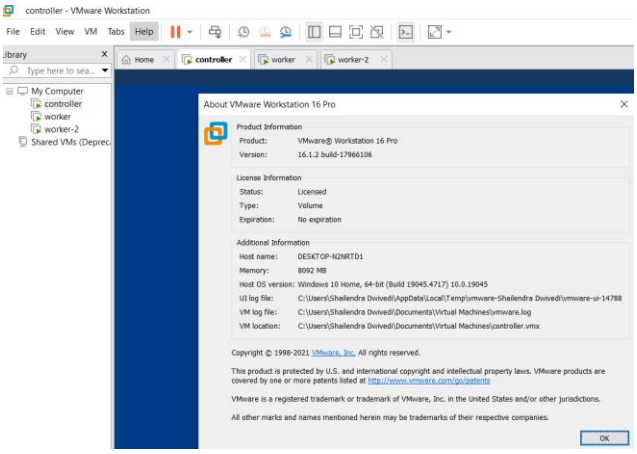


Figure 3: VMware Workstation hypervisor Type-2

Set Up Ansible Controller Node:
Ensure you have a machine (local or remote) where Ansible is installed. This will be your control node.

Setup worker Nodes:
Set up one or more remote servers that Ansible will manage. These servers should be reachable via SSH and have Python installed.

Create an Inventory File:
The inventory file lists the hosts Ansible will manage. Save this file as inventory

[root@controller ~] # vi inventory

[nfsserver]
server1 ansible_host=192.168.29.22
server2 ansible_host=192.168.29.48

```
[root@controller ~]# cat inventory
[nfsserver]
192.168.29.22
192.168.29.48
[root@controller ~]#
```

Figure 4: Worker's node IP configuration in Inventory file

Table1: Specification of Servers

Sr. No.	*Devices*	*RAM,CPU,HD*	*OS Version*	*IP address*
1	Controller Node	2Gb,1Core, 20Gb(SCSi)	Centos_v7.9	192.168.29.132
2	Worker Node-1	1Gb,1Core, 20Gb(SCSi)	Centos_v7.9	192.168.29.22
3	Worker Node-2	1Gb,1Core, 20Gb(SCSi)	Centos_v7.9	192.168.29.48

Controller Node:

```
[root@controller ~]# hostname;cat /etc/redhat-release;free -g;df -Th;lscpu |grep -i core
controller.example.com
CentOS Linux release 7.9.2009 (Core)
              total        used        free      shared  buff/cache   available
Mem:              1           0           0           0           0           0
Swap:             1           0           1
Filesystem              Type      Size  Used Avail Use% Mounted on
devtmpfs                devtmpfs  974M     0  974M   0% /dev
tmpfs                   tmpfs     991M     0  991M   0% /dev/shm
tmpfs                   tmpfs     991M   11M  980M   2% /run
tmpfs                   tmpfs     991M     0  991M   0% /sys/fs/cgroup
/dev/mapper/centos-root xfs        17G  4.3G   13G  26% /
/dev/sda1               xfs      1014M  172M  843M  17% /boot
tmpfs                   tmpfs     199M   28K  199M   1% /run/user/0
/dev/sr0                iso9660   4.4G  4.4G     0 100% /run/media/root/CentOS 7 x86_64
Thread(s) per core:    1
Core(s) per socket:    1
Model name:            Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
[root@controller ~]#
```

Figure 5: Specification of Controller Node

Worker Node-1:

```
[root@worker-1 ~]# hostname;cat /etc/redhat-release;free -g;df -Th;lscpu |grep -i core
worker-1.example.com
CentOS Linux release 7.9.2009 (Core)
              total        used        free      shared  buff/cache   available
Mem:              0           0           0           0           0           0
Swap:             1           0           1
Filesystem              Type      Size  Used Avail Use% Mounted on
devtmpfs                devtmpfs  470M     0  470M   0% /dev
tmpfs                   tmpfs     487M     0  487M   0% /dev/shm
tmpfs                   tmpfs     487M  8.6M  478M   2% /run
tmpfs                   tmpfs     487M     0  487M   0% /sys/fs/cgroup
/dev/mapper/centos-root xfs        17G  4.2G   13G  25% /
/dev/sda1               xfs      1014M  172M  843M  17% /boot
tmpfs                   tmpfs     98M   32K   98M   1% /run/user/0
/dev/sr0                iso9660   4.4G  4.4G     0 100% /run/media/root/CentOS 7 x86_64
Thread(s) per core:    1
Core(s) per socket:    1
Model name:            Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
[root@worker-1 ~]#
```

Figure 6: Specification of Worker Node-1 Node

Worker Node-2:

```
[root@worker-2 ~]# hostname;cat /etc/redhat-release;free -g;df -Th;lscpu |grep -i core
worker-2.example.com
CentOS Linux release 7.9.2009 (Core)
              total        used        free      shared  buff/cache   available
Mem:              0           0           0           0           0           0
Swap:             1           0           1
Filesystem              Type      Size  Used Avail Use% Mounted on
devtmpfs                devtmpfs  470M     0  470M   0% /dev
tmpfs                   tmpfs     487M     0  487M   0% /dev/shm
tmpfs                   tmpfs     487M  8.6M  478M   2% /run
tmpfs                   tmpfs     487M     0  487M   0% /sys/fs/cgroup
/dev/mapper/centos-root xfs        17G  4.2G   13G  25% /
/dev/sda1               xfs      1014M  172M  843M  17% /boot
tmpfs                   tmpfs     98M   28K   98M   1% /run/user/0
/dev/sr0                iso9660   4.4G  4.4G     0 100% /run/media/root/CentOS 7 x86_64
Thread(s) per core:    1
Core(s) per socket:    1
Model name:            Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
[root@worker-2 ~]#
```

Figure 7: Specification of Worker Node-2 Node

Create an Ansible Playbook: Write a playbook to install and start `nfs` on the remote servers. Save this file as `install_nfs.yaml`.
Package name-: nfs-utils-1.3.0-0.68.el7.x86_64

For creation of anisible playbook to install nfs server, first we write an yaml, below file yaml language to install & start the nfs service in both worker node:

```
[root@controller ~]# cat install_nfs.yaml
---
- name: Install and configure nfs Server
  hosts: nfsserver
  become: yes
  tasks:
    - name: Ensure nfs is installed
      yum:
        name: nfs-utils
        state: present

    - name: Ensure nfs is started and enabled
      service:
        name: nfs-utils
        state: started
        enabled: yes
[root@controller ~]#
```

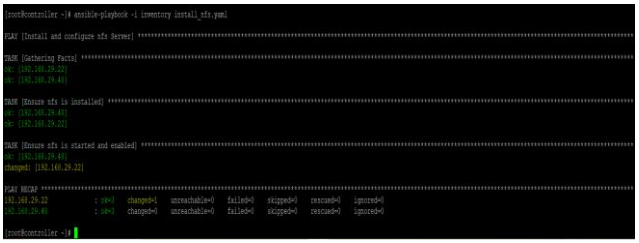
Figure 8: yaml script for Installation of NFS.

After written yaml, we need to execute this yaml file to deploy nfs server on both workers.

Run the Playbook:
Use the Ansible command line to run the playbook against the inventory.

```
[root@controller ~]# hostname;ls -ltrh
controller.example.com
total 16K
-rw-------. 1 root root 1.5K Jun  9 22:08 anaconda-ks.cfg
-rw-r--r--. 1 root root 1.7K Jun  9 22:16 initial-setup-ks.cfg
drwxr-xr-x. 2 root root    6 Jun  9 22:19 Desktop
drwxr-xr-x. 2 root root    6 Jun  9 22:19 Videos
drwxr-xr-x. 2 root root    6 Jun  9 22:19 Templates
drwxr-xr-x. 2 root root    6 Jun  9 22:19 Public
drwxr-xr-x. 2 root root    6 Jun  9 22:19 Pictures
drwxr-xr-x. 2 root root    6 Jun  9 22:19 Music
drwxr-xr-x. 2 root root    6 Jun  9 22:19 Downloads
drwxr-xr-x. 2 root root    6 Jun  9 22:19 Documents
-rw-r--r--. 1 root root   40 Aug  2 23:06 inventory
-rw-r--r--. 1 root root  311 Aug  2 23:08 install_nfs.yaml
[root@controller ~]#
```

[root@controller ~]# ansible-playbook -i inventory install_nfs.yaml



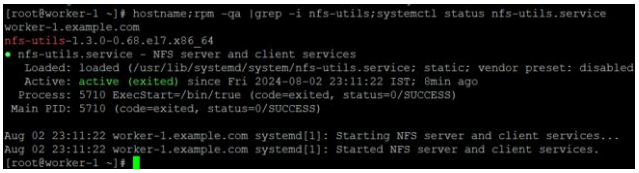
This output indicates that the NFS package was installed and the service was started and enabled on both servers.

By following these steps, you can simulate the installation and configuration of the NFS Server using Ansible, demonstrating its power and simplicity in automating system administration tasks.

6. Verify Installation & Service status of NFS

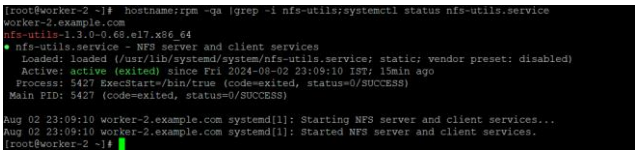
After the playbook runs, verify that NFSd is installed and running on the managed nodes by SSHing into the nodes and checking the status of the nfs service.

Worker-1: Verify the status of nfs service in worker-1 through ssh



From above image it's clear that nfs service successfully installed from controller node using ansible-playbook yaml and services also started on worker node-1.

Worker-2: Verify the status of nfs service in worker-2 through ssh



From above image it's clear that nfs service successfully installed from controller node using ansible-playbook yaml and services also started on worker node-2

7. Conclusion

The automation of NFS Server (NFS) installation and configuration using Ansible showcases the powerful capabilities of Ansible in managing complex IT infrastructure tasks efficiently and reliably. Through a well-structured playbook, Ansible can ensure the consistent deployment of NFS services across multiple servers, significantly reducing manual effort and the potential for human error.

By leveraging Ansible's agentless architecture, simple YAML syntax, and idempotent operations, administrators can automate repetitive tasks, maintain uniform configurations, and achieve a high degree of scalability in managing distributed systems. The step-by-step approach, from setting up the control node and managed nodes to creating and running the playbook, demonstrates a clear and effective method for implementing NFS services in diverse environments.

Incorporating Ansible for the deployment of NFS servers not only enhances operational efficiency but also promotes best practices in infrastructure as code (IaC), fostering a more agile and resilient IT ecosystem. This approach ensures that NFS services are consistently configured and readily available, supporting seamless file sharing and collaboration across the network.

In conclusion, Ansible proves to be an invaluable tool for automating the installation and management of NFS servers, empowering organizations to streamline their IT operations and achieve greater consistency, reliability, and scalability in their infrastructure management practices.

References

[1] Ramandeep Singh, Dr. Ravindra Kumar Purwar, 2019, Cloud Automation with Configuration Managementusing CHEF Tool, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 08, Issue 04 (April – 2019)

[2] Masek, Pavel & Štůsek, Martin & Krejčí, Jan & Zeman, Krystof & Pokorny, Jiri & Kudlacek, Marek. (2018). Unleashing Full Potential of Ansible Framework: University Labs Administration. Proceedings of the XXth Conference of Open Innovations Association FRUCT. 426. 10.23919/FRUCT.2018.8468270.

[3] Kumari, Priti, and Parmeet Kaur. "A survey of fault tolerance in cloud computing."Journal of King Saud University-Computer and Information Sciences 33.10 (2021):.

[4] Rao, Umesh Hodeghatta, and Umesha Nayak. "Data backups and cloud computing." The InfoSec Handbook. Apress, Berkeley, CA, 2014. 263-288.

[5] Gentile, Ugo, and Luigi Serio. "Survey on international standards and best practices for patch management of complex industrial control systems: the critical infrastructure of particle accelerators case study." International Journal of Critical Computer-Based Systems 9.1-2 (2019): 115-132.

[6] Tomarchio, Orazio, Domenico Calcaterra, and Giuseppe Di Modica. "Cloud resource orchestration in the multi-cloud landscape: a systematic review of existing frameworks." Journal of Cloud Computing 9.1 (2020): 1-24.

[7] Weerasiri, Denis, et al. "A taxonomy and survey of cloud resource orchestration techniques." ACM Computing Surveys (CSUR) 50.2 (2017): 1-41