Assignment: Configuration Management with Ansible and Puppet

Introduction

This assignment demonstrates the implementation of two major configuration management tools: **Ansible** and **Puppet**. The objective is to understand their architecture, components, and practical applications in automating system administration tasks.

Assignment Objectives:

- Implement Ansible for web server (nginx) deployment and firewall configuration
- Implement Puppet for automated user management
- Compare different configuration management tools
- Document the complete implementation process with evidence

Part 1: Ansible - Nginx Installation and Firewall Configuration

1.1 Overview

Ansible is an agentless configuration management tool that uses SSH for communication. This section demonstrates using Ansible to automate the installation and configuration of nginx web server along with firewall setup.

Key Components Used:

- **Inventory**: Defines target hosts
- **Modules**: Pre-built functionality (apt, ufw, service, etc.)
- **Playbooks**: YAML files containing automation tasks
- **Roles**: Reusable automation components

1.2 Environment Details

Control Machine:

- **IP Address**: 192.168.56.10
- **Hostname**: control
- **Role**: Ansible control node

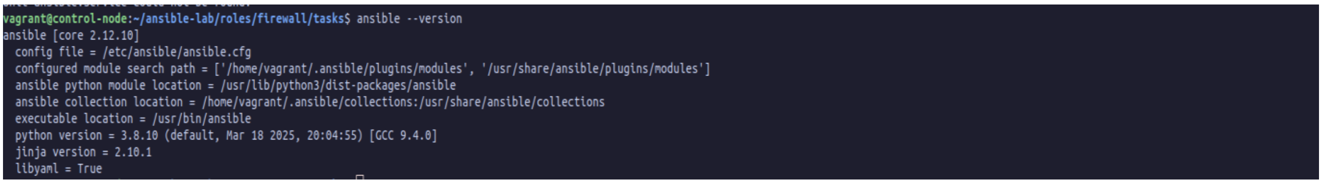
Target Machine:

- **IP Address**: 192.168.56.11
- **Hostname**: node1
- **Role**: Web server (nginx installation target)

1.3 Implementation Steps

Step 1: Ansible Installation and Setup

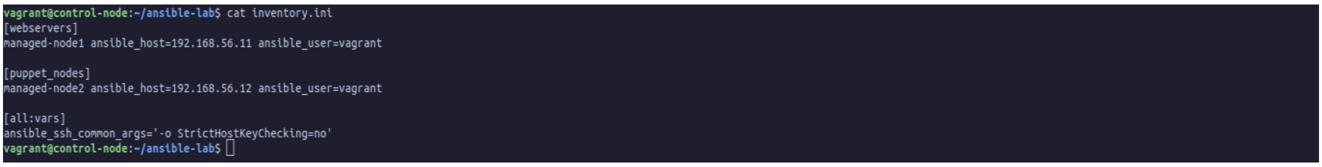
Installation Process: The Ansible control node was set up with the latest version of Ansible. SSH connectivity was established between control and target nodes.



```
vagrant@control-node:~/ansible-lab/roles/firewall/tasks$ ansible --version
ansible [core 2.12.10]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/vagrant/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/vagrant/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.8.10 (default, Mar 18 2025, 20:04:55) [GCC 9.4.0]
  jinja version = 2.10.1
  libyaml = True
```

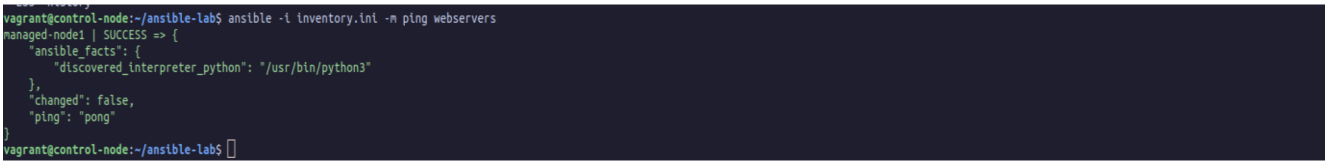
Step 2: Inventory Configuration

Inventory File Structure: The inventory file defines the target hosts and their connection parameters.



```
vagrant@control-node:~/ansible-lab$ cat inventory.ini
[webservers]
managed-node1 ansible_host=192.168.56.11 ansible_user=vagrant

[puppet_nodes]
managed-node2 ansible_host=192.168.56.12 ansible_user=vagrant

[all:vars]
ansible_ssh_common_args='-o StrictHostKeyChecking=no'
vagrant@control-node:~/ansible-lab$
```

Connectivity Test: Verified connectivity to all hosts in the inventory.



```
vagrant@control-node:~/ansible-lab$ ansible -i inventory.ini -m ping webservers
managed-node1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
vagrant@control-node:~/ansible-lab$
```

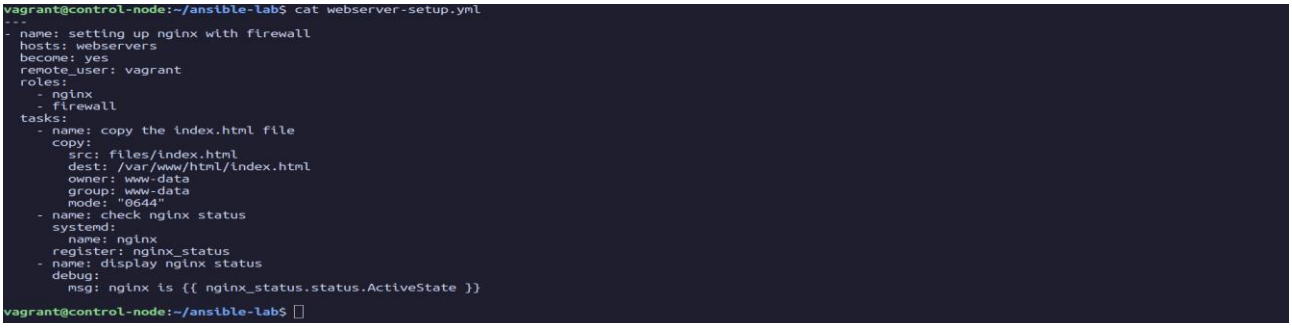
Step 3: Ansible Modules Used

The following Ansible modules were utilized in this implementation:

1. **apt module**: Package management
2. **ufw module**: Firewall configuration
3. **service module**: Service management
4. **file module**: File and directory operations
5. **copy module**: File transfer

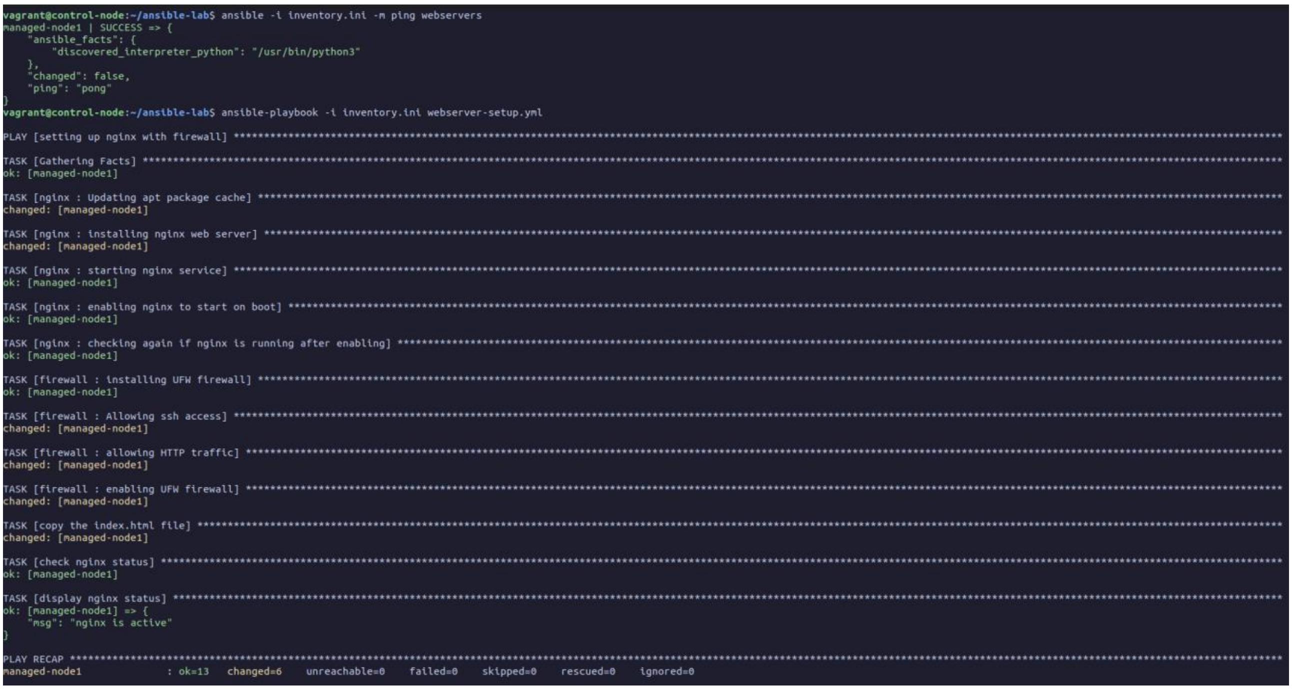
Step 4: Playbook Development

Playbook Structure: A comprehensive playbook was created to handle nginx installation, configuration, and firewall setup.



```
vagrant@control-node:~/ansible-lab$ cat webserver-setup.yml
---
- name: setting up nginx with firewall
  hosts: webservers
  become: yes
  remote_user: vagrant
  roles:
    - nginx
    - firewall
  tasks:
    - name: copy the index.html file
      copy:
        src: files/index.html
        dest: /var/www/html/index.html
        owner: www-data
        group: www-data
        mode: "0644"
    - name: check nginx status
      systemd:
        name: nginx
      register: nginx_status
    - name: display nginx status
      debug:
        msg: nginx is {{ nginx_status.status.ActiveState }}

vagrant@control-node:~/ansible-lab$ 
```

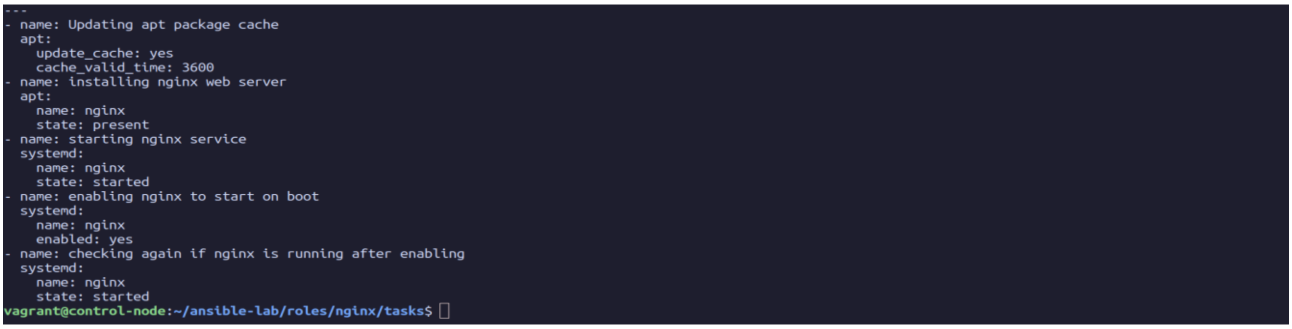
Playbook Execution: The playbook was executed against the target hosts.



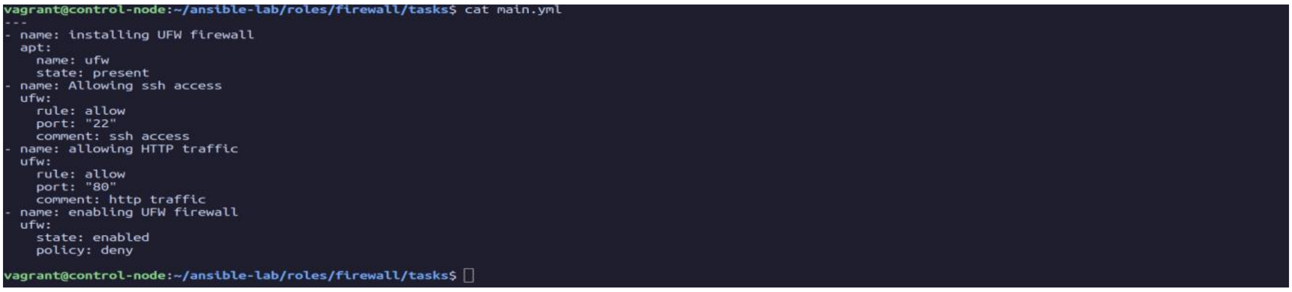
```
vagrant@control-node:~/ansible-lab$ ansible -i inventory.ini -m ping webservers
managed-node1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
vagrant@control-node:~/ansible-lab$ ansible-playbook -i inventory.ini webserver-setup.yml

PLAY [setting up nginx with firewall] ******************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************
ok: [managed-node1]

TASK [nginx : Updating apt package cache] **************************************************************************************************************************
changed: [managed-node1]

TASK [nginx : installing nginx web server] *************************************************************************************************************************
changed: [managed-node1]

TASK [nginx : starting nginx service] ******************************************************************************************************************************
ok: [managed-node1]

TASK [nginx : enabling nginx to start on boot] *********************************************************************************************************************
ok: [managed-node1]

TASK [nginx : checking again if nginx is running after enabling] ***************************************************************************************************
ok: [managed-node1]

TASK [firewall : installing UFW firewall] **************************************************************************************************************************
ok: [managed-node1]

TASK [firewall : Allowing ssh access] ******************************************************************************************************************************
changed: [managed-node1]

TASK [firewall : allowing HTTP traffic] ****************************************************************************************************************************
changed: [managed-node1]

TASK [firewall : enabling UFW firewall] ****************************************************************************************************************************
changed: [managed-node1]

TASK [copy the index.html file] ************************************************************************************************************************************
changed: [managed-node1]

TASK [check nginx status] ******************************************************************************************************************************************
ok: [managed-node1]

TASK [display nginx status] ****************************************************************************************************************************************
ok: [managed-node1] => {
    "msg": "nginx is active"
}

PLAY RECAP *********************************************************************************************************************************************************
managed-node1              : ok=13   changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Step 5: Roles Implementation

Role Structure: Ansible roles were created for better organization and reusability.



```
---
- name: Updating apt package cache
  apt:
    update_cache: yes
    cache_valid_time: 3600
- name: installing nginx web server
  apt:
    name: nginx
    state: present
- name: starting nginx service
  systemd:
    name: nginx
    state: started
- name: enabling nginx to start on boot
  systemd:
    name: nginx
    enabled: yes
- name: checking again if nginx is running after enabling
  systemd:
    name: nginx
    state: started
vagrant@control-node:~/ansible-lab/roles/nginx/tasks$ 
```

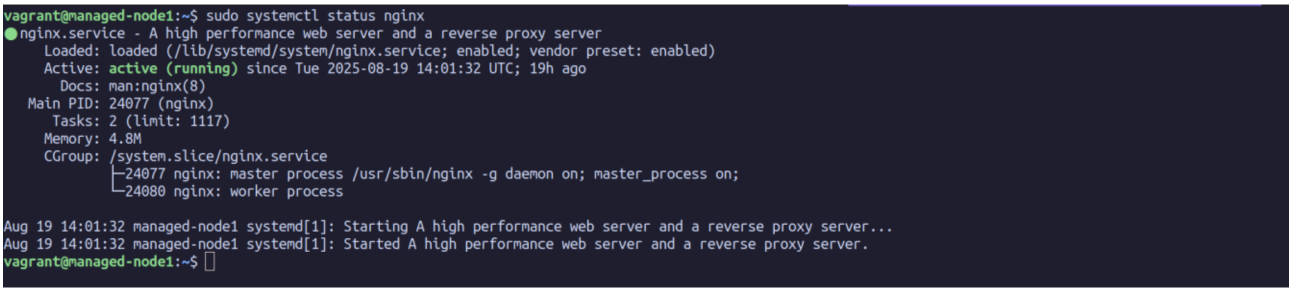


```
vagrant@control-node:~/ansible-lab/roles/firewall/tasks$ cat main.yml
---
- name: installing UFW firewall
  apt:
    name: ufw
    state: present
- name: Allowing ssh access
  ufw:
    rule: allow
    port: "22"
    comment: ssh access
- name: allowing HTTP traffic
  ufw:
    rule: allow
    port: "80"
    comment: http traffic
- name: enabling UFW firewall
  ufw:
    state: enabled
    policy: deny

vagrant@control-node:~/ansible-lab/roles/firewall/tasks$ 
```

1.4 Results and Verification

Nginx Installation Verification

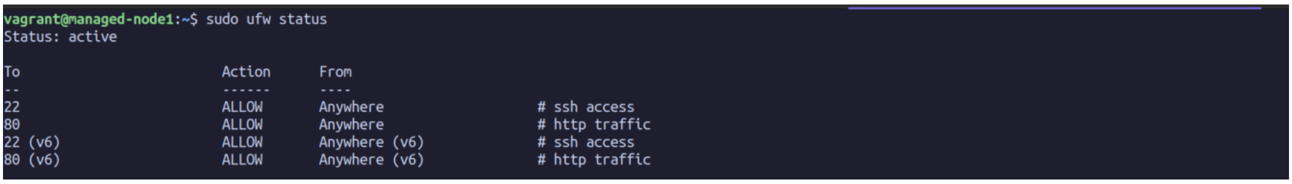
Service Status: Verified that nginx service is running and enabled.



```
vagrant@managed-node1:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2025-08-19 14:01:32 UTC; 19h ago
       Docs: man:nginx(8)
   Main PID: 24077 (nginx)
      Tasks: 2 (limit: 1117)
     Memory: 4.8M
     CGroup: /system.slice/nginx.service
             ├─24077 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             └─24080 nginx: worker process

Aug 19 14:01:32 managed-node1 systemd[1]: Starting A high performance web server and a reverse proxy server...
Aug 19 14:01:32 managed-node1 systemd[1]: Started A high performance web server and a reverse proxy server.
vagrant@managed-node1:~$ 
```

Firewall Configuration Verification

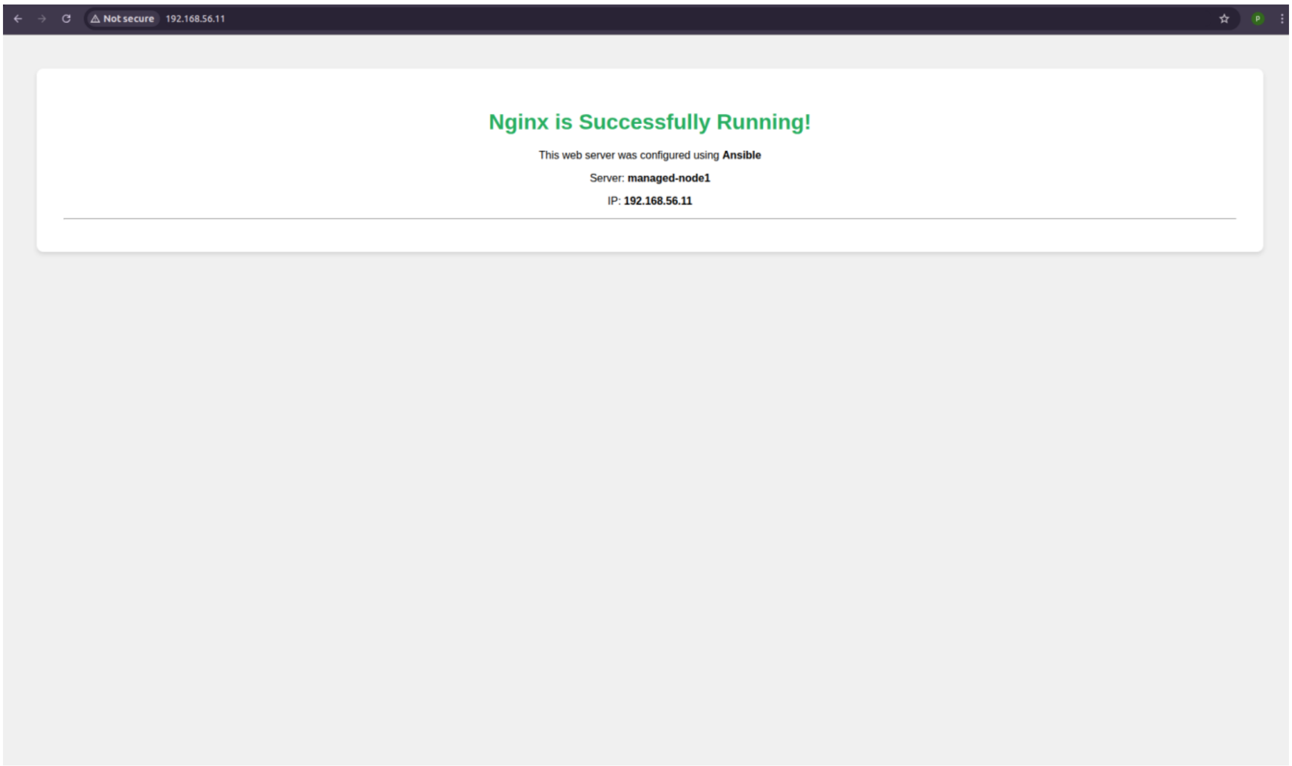
UFW Status: Confirmed firewall rules are properly configured.



```
vagrant@managed-node1:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere                  # ssh access
80                         ALLOW       Anywhere                  # http traffic
22 (v6)                    ALLOW       Anywhere (v6)             # ssh access
80 (v6)                    ALLOW       Anywhere (v6)             # http traffic
```

Web Server Accessibility

HTTP Access Test: Verified nginx is serving pages correctly.



Nginx is Successfully Running!

This web server was configured using **Ansible**

Server: **managed-node1**

IP: **192.168.56.11**

Part 2: Puppet - User Management

2.1 Overview

Puppet is a declarative configuration management tool that uses a master-agent architecture. This section demonstrates using Puppet to automate user creation and management across multiple nodes.

Key Components Used:

- **Manifests**: Puppet code files describing desired system state
- **Modules**: Reusable collections of manifests, templates, and files
- **Master-Agent Architecture**: Centralized configuration management

2.2 Environment Details

Puppet Master (Control Node):

- **IP Address**: 192.168.56.10
- **Hostname**: control
- **Role**: Puppet server/master

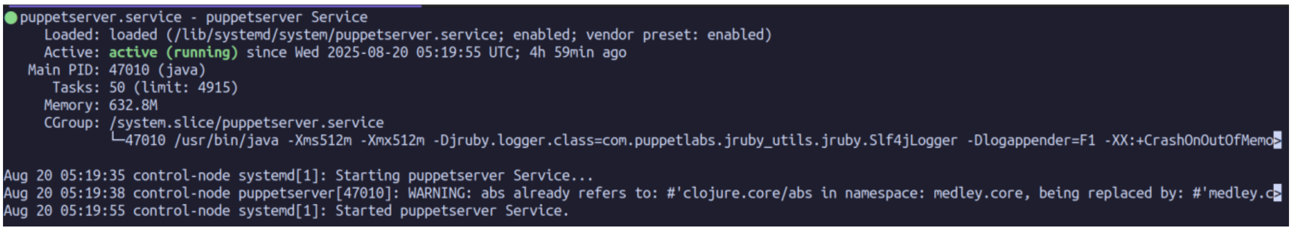
Puppet Agent (Managed Node):

- **IP Address**: 192.168.56.12
- **Hostname**: node2
- **Role**: Managed node for user creation

2.3 Implementation Steps

Step 1: Puppet Master Installation

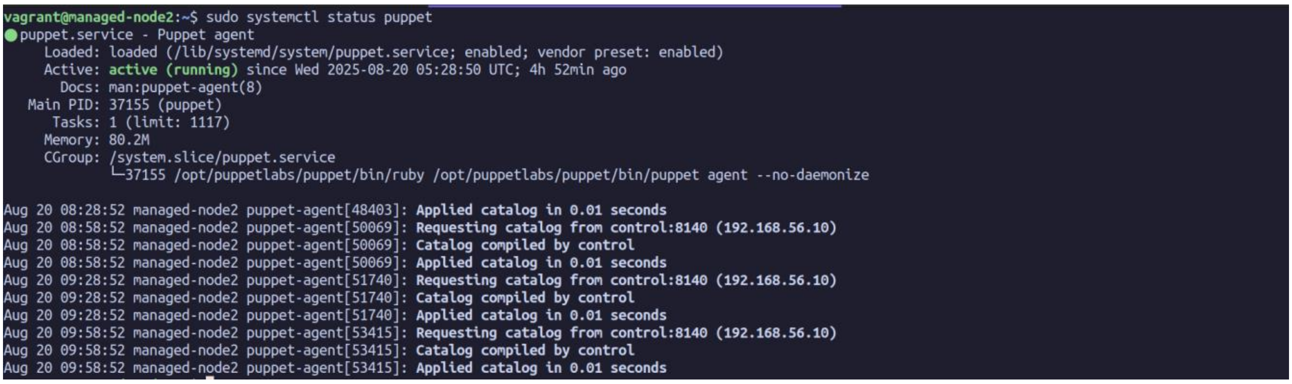
Installation Process: Puppet Server was installed and configured on the control node.



```
●puppetserver.service - puppetserver Service
     Loaded: loaded (/lib/systemd/system/puppetserver.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2025-08-20 05:19:55 UTC; 4h 59min ago
   Main PID: 47010 (java)
      Tasks: 50 (limit: 4915)
     Memory: 632.8M
     CGroup: /system.slice/puppetserver.service
             └─47010 /usr/bin/java -Xms512m -Xmx512m -Djruby.logger.class=com.puppetlabs.jruby_utils.jruby.Slf4jLogger -Dlogappender=F1 -XX:+CrashOnOutOfMemo>

Aug 20 05:19:35 control-node systemd[1]: Starting puppetserver Service...
Aug 20 05:19:38 control-node puppetserver[47010]: WARNING: abs already refers to: #'clojure.core/abs in namespace: medley.core, being replaced by: #'medley.c>
Aug 20 05:19:55 control-node systemd[1]: Started puppetserver Service.
```

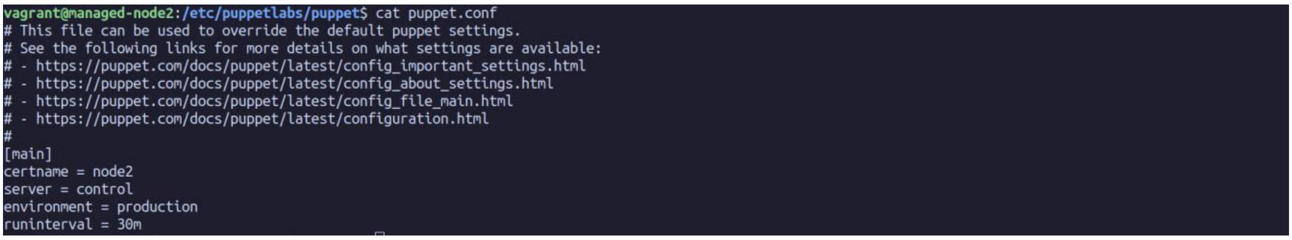
Step 2: Puppet Agent Installation

Agent Installation: Puppet agent was installed on the target node.



```
vagrant@managed-node2:~$ sudo systemctl status puppet
●puppet.service - Puppet agent
     Loaded: loaded (/lib/systemd/system/puppet.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2025-08-20 05:28:50 UTC; 4h 52min ago
       Docs: man:puppet-agent(8)
   Main PID: 37155 (puppet)
      Tasks: 1 (limit: 1117)
     Memory: 80.2M
     CGroup: /system.slice/puppet.service
             └─37155 /opt/puppetlabs/puppet/bin/ruby /opt/puppetlabs/puppet/bin/puppet agent --no-daemonize

Aug 20 08:28:52 managed-node2 puppet-agent[48403]: Applied catalog in 0.01 seconds
Aug 20 08:58:52 managed-node2 puppet-agent[50069]: Requesting catalog from control:8140 (192.168.56.10)
Aug 20 08:58:52 managed-node2 puppet-agent[50069]: Catalog compiled by control
Aug 20 08:58:52 managed-node2 puppet-agent[50069]: Applied catalog in 0.01 seconds
Aug 20 09:28:52 managed-node2 puppet-agent[51740]: Requesting catalog from control:8140 (192.168.56.10)
Aug 20 09:28:52 managed-node2 puppet-agent[51740]: Catalog compiled by control
Aug 20 09:28:52 managed-node2 puppet-agent[51740]: Applied catalog in 0.01 seconds
Aug 20 09:58:52 managed-node2 puppet-agent[53415]: Requesting catalog from control:8140 (192.168.56.10)
Aug 20 09:58:52 managed-node2 puppet-agent[53415]: Catalog compiled by control
Aug 20 09:58:52 managed-node2 puppet-agent[53415]: Applied catalog in 0.01 seconds
```

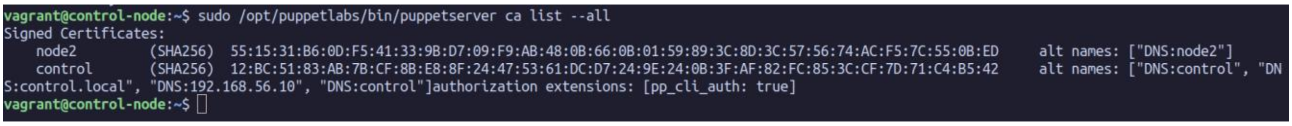
Agent Configuration: Configured agent to connect to puppet master.



```
vagrant@managed-node2:/etc/puppetlabs/puppet$ cat puppet.conf
# This file can be used to override the default puppet settings.
# See the following links for more details on what settings are available:
# - https://puppet.com/docs/puppet/latest/config_important_settings.html
# - https://puppet.com/docs/puppet/latest/config_about_settings.html
# - https://puppet.com/docs/puppet/latest/config_file_main.html
# - https://puppet.com/docs/puppet/latest/configuration.html
#
[main]
certname = node2
server = control
environment = production
runinterval = 30m
```

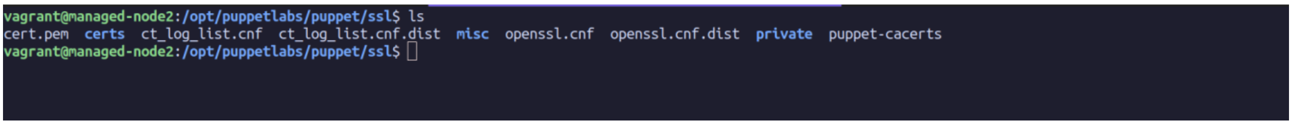
Step 3: Certificate Management

Certificate Signing: Master signed the agent's certificate.



```
vagrant@control-node:~$ sudo /opt/puppetlabs/bin/puppetserver ca list --all
Signed Certificates:
    node2        (SHA256)  55:15:31:B6:0D:F5:41:33:9B:D7:09:F9:AB:48:0B:66:0B:01:59:89:3C:8D:3C:57:56:74:AC:F5:7C:55:0B:ED     alt names: ["DNS:node2"]
    control      (SHA256)  12:BC:51:83:AB:7B:CF:8B:E8:8F:24:47:53:61:DC:D7:24:9E:24:0B:3F:AF:82:FC:85:3C:CF:7D:71:C4:B5:42     alt names: ["DNS:control", "DN
S:control.local", "DNS:192.168.56.10", "DNS:control"]authorization extensions: [pp_cli_auth: true]
vagrant@control-node:~$
```

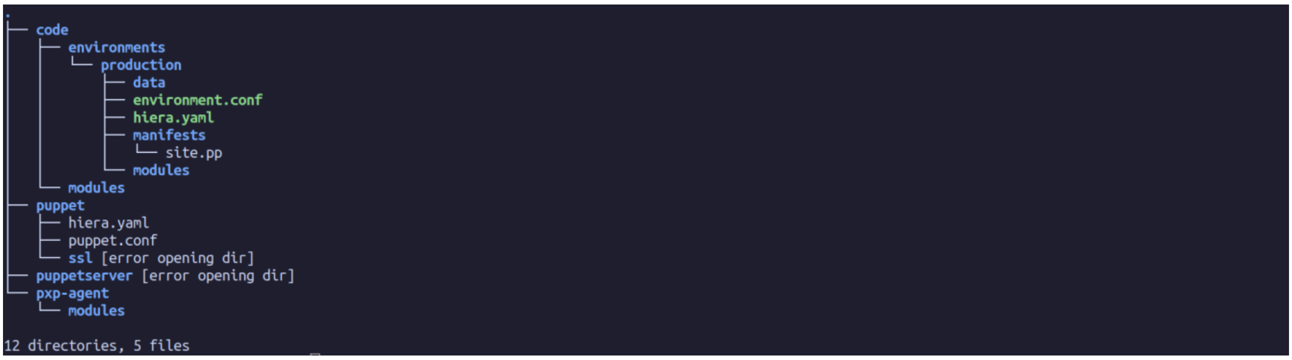
Certificate Verification: Verified successful certificate exchange.(cert.pem file exists)



```
vagrant@managed-node2:/opt/puppetlabs/puppet/ssl$ ls
cert.pem  certs  ct_log_list.cnf  ct_log_list.cnf.dist  misc  openssl.cnf  openssl.cnf.dist  private  puppet-cacerts
vagrant@managed-node2:/opt/puppetlabs/puppet/ssl$
```

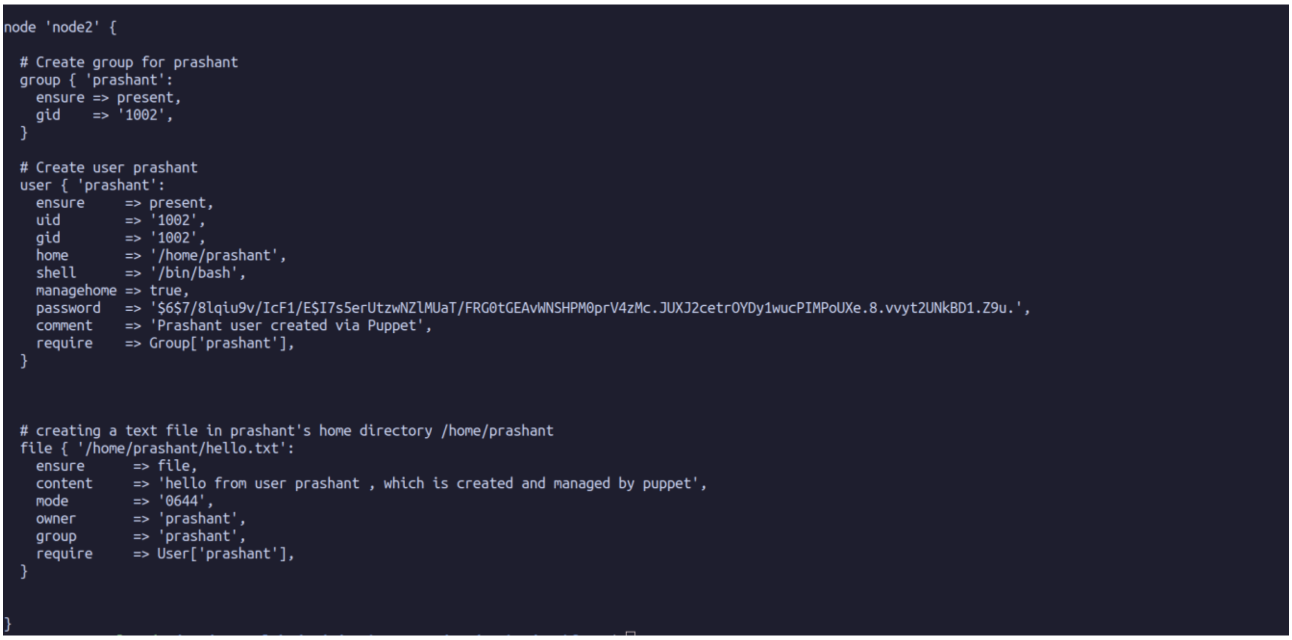
Step 4: Manifest Development

File Structure: Puppet code organization and file structure.



```
.
├── code
│   ├── environments
│   │   └── production
│   │       ├── data
│   │       ├── environment.conf
│   │       ├── hiera.yaml
│   │       ├── manifests
│   │       │   └── site.pp
│   │       └── modules
│   └── modules
├── puppet
│   ├── hiera.yaml
│   ├── puppet.conf
│   └── ssl [error opening dir]
├── puppetserver [error opening dir]
└── pxp-agent
    └── modules

12 directories, 5 files
```

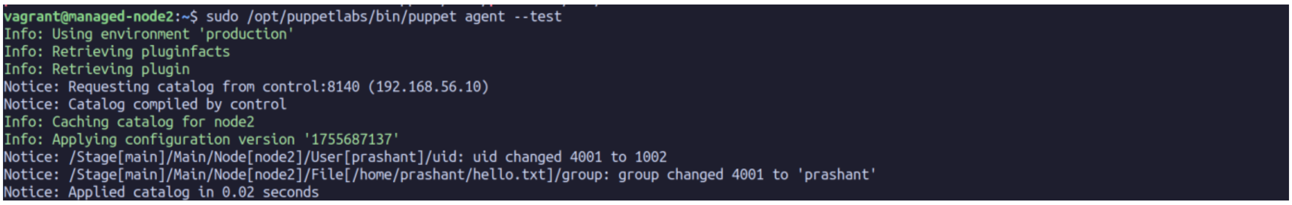
Main Manifest: Created site.pp manifest for user management.



```
node 'node2' {

  # Create group for prashant
  group { 'prashant':
    ensure => present,
    gid    => '1002',
  }

  # Create user prashant
  user { 'prashant':
    ensure     => present,
    uid        => '1002',
    gid        => '1002',
    home       => '/home/prashant',
    shell      => '/bin/bash',
    managehome => true,
    password   => '$6$7/8lqiu9v/IcF1/E$I7s5erUtzwNZlMUaT/FRG0tGEAvWNSHPM0prV4zMc.JUXJ2cetrOYDy1wucPIMPoUXe.8.vvyt2UNkBD1.Z9u.',
    comment    => 'Prashant user created via Puppet',
    require    => Group['prashant'],
  }


  # creating a text file in prashant's home directory /home/prashant
  file { '/home/prashant/hello.txt':
    ensure      => file,
    content     => 'hello from user prashant , which is created and managed by puppet',
    mode        => '0644',
    owner       => 'prashant',
    group       => 'prashant',
    require     => User['prashant'],
  }

}
```

Step 5: Puppet Run and Deployment

Initial Puppet Run: Applied configuration to managed node.

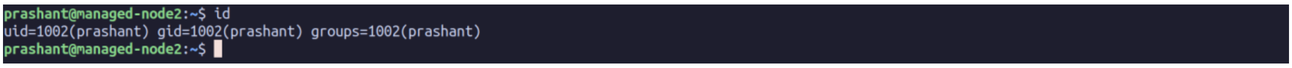


```
vagrant@managed-node2:~$ sudo /opt/puppetlabs/bin/puppet agent --test
Info: Using environment 'production'
Info: Retrieving pluginfacts
Info: Retrieving plugin
Notice: Requesting catalog from control:8140 (192.168.56.10)
Notice: Catalog compiled by control
Info: Caching catalog for node2
Info: Applying configuration version '1755687137'
Notice: /Stage[main]/Main/Node[node2]/User[prashant]/uid: uid changed 4001 to 1002
Notice: /Stage[main]/Main/Node[node2]/File[/home/prashant/hello.txt]/group: group changed 4001 to 'prashant'
Notice: Applied catalog in 0.02 seconds
```

2.4 Results and Verification

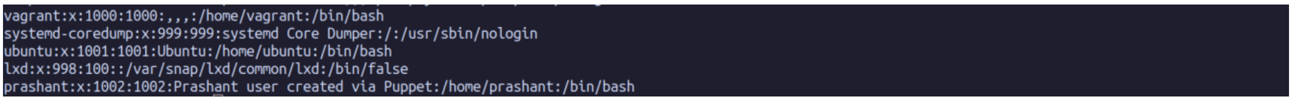
User Creation Verification

User Existence Check: Verified user was created successfully.



```
prashant@managed-node2:~$ id
uid=1002(prashant) gid=1002(prashant) groups=1002(prashant)
prashant@managed-node2:~$
```

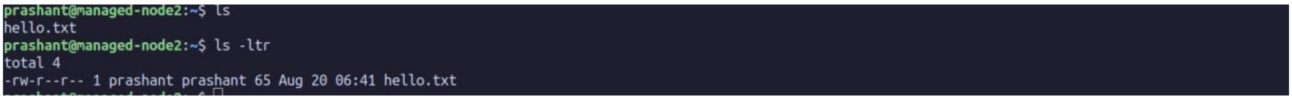
Password File Entry: Confirmed user entry in system password file.



```
vagrant:x:1000:1000:,,,:/home/vagrant:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
ubuntu:x:1001:1001:Ubuntu:/home/ubuntu:/bin/bash
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
prashant:x:1002:1002:Prashant user created via Puppet:/home/prashant:/bin/bash
```

Home Directory Verification

Directory Creation: Verified home directory was created with proper permissions and hello.txt file



```
prashant@managed-node2:~$ ls
hello.txt
prashant@managed-node2:~$ ls -ltr
total 4
-rw-r--r-- 1 prashant prashant 65 Aug 20 06:41 hello.txt
```

Comparison of Configuration Management Tools

3.1 Tool Comparison Matrix

Feature	Ansible	Puppet	Chef
Architecture	Agentless	Master-Agent	Master-Agent
Language	YAML	Ruby DSL	Ruby
Learning Curve	Easy	Moderate	Steep
Communication	SSH	HTTPS/SSL	HTTPS/SSL
Configuration	Push-based	Pull-based	Pull-based
Idempotency	Yes	Yes	Yes

3.2 Best Use Cases

Ansible:

- **Ideal for**: Simple automation, ad-hoc tasks, multi-vendor environments
- **Strengths**: Easy to learn, agentless, great for orchestration
- **Weaknesses**: Performance with large infrastructures

Puppet:

- **Ideal for**: Large-scale infrastructure, compliance management, complex configurations
- **Strengths**: Mature ecosystem, excellent reporting, strong community
- **Weaknesses**: Learning curve, requires dedicated infrastructure

Chef:

- **Ideal for**: Complex application deployment, integration with development workflows
- **Strengths**: Flexible, powerful testing framework, cloud integration

- **Weaknesses**: Steep learning curve, complex setup