Create and manage Puppet environments to handle different configurations

Table of Contents

Introduction

Puppet environments are a powerful way to manage different configurations across multiple nodes. For example, you might want to apply different configurations for development, staging, and production environments. By using environments, you can manage these configurations separately without conflicts.

Problem Statement

Managing different configurations for multiple stages (like development, testing, and production) within a single Puppet setup can be error-prone and difficult to maintain. Puppet environments allow for clean separation of configuration code based on the target environment.

Prerequisites

Completion of all previous lab guides (up to Lab Guide-05) is required before proceeding with Lab Guide-06.

Software Requirements

- Puppet 3.8.7
- Puppet Master and Agent properly installed
- A file system that supports multiple directories for different environments

Hardware Requirements

- Puppet Master: Minimum 1GB RAM, 2 CPUs, 10GB Disk
- Puppet Agent: Minimum 512MB RAM, 1 CPU, 5GB Disk

Implementation Steps

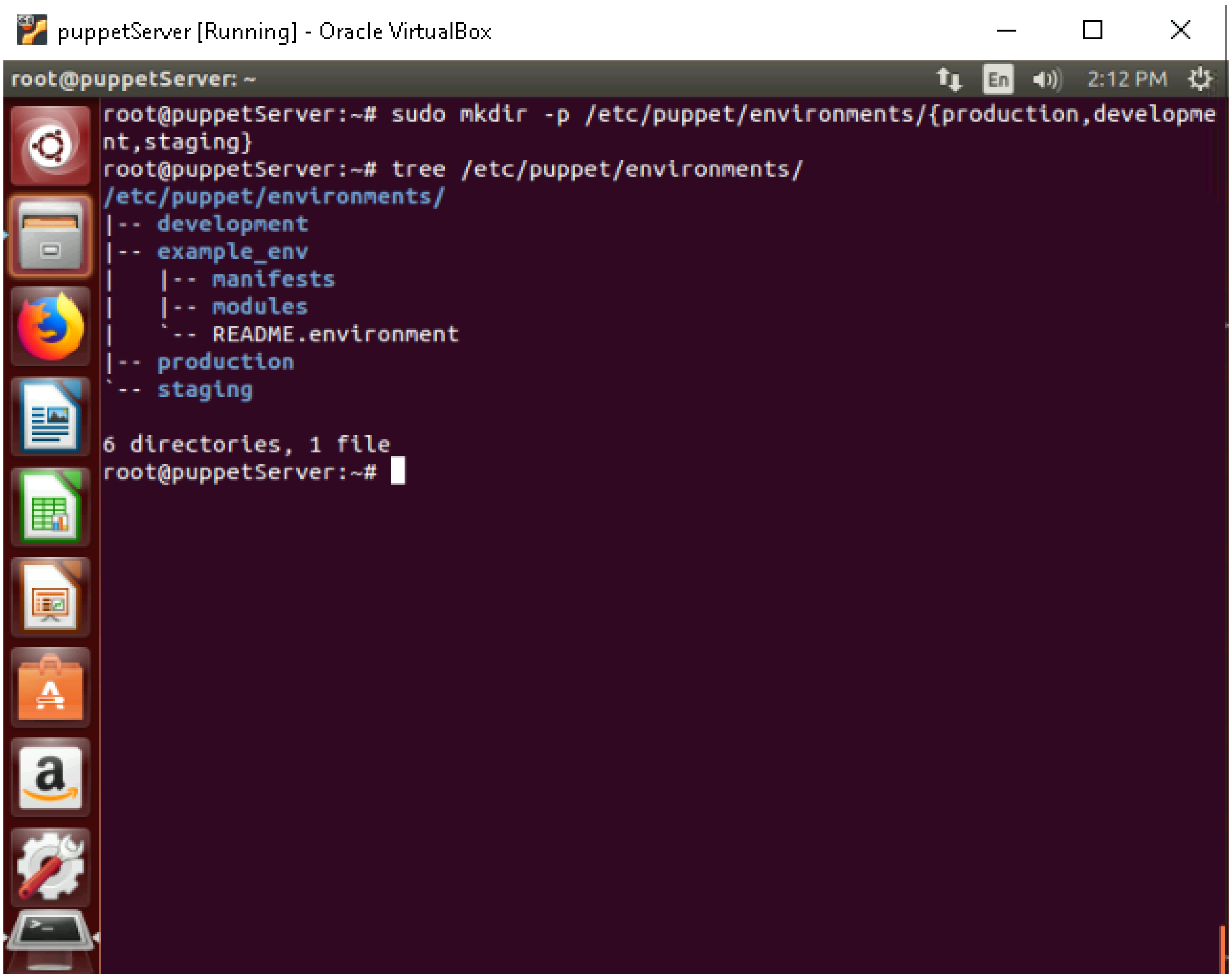
Step 1: Set Up Puppet Environment Directory Structure

1. **Create a Directory for Environments**:

   The Puppet Master needs a dedicated directory for environments, where each environment can have its own configuration (manifests, modules, etc.). By default, this directory is `/etc/puppet/environments/`.

   To create and set up environments:

   ```
   sudo mkdir -p /etc/puppet/environments/{production,development,staging}
   ```

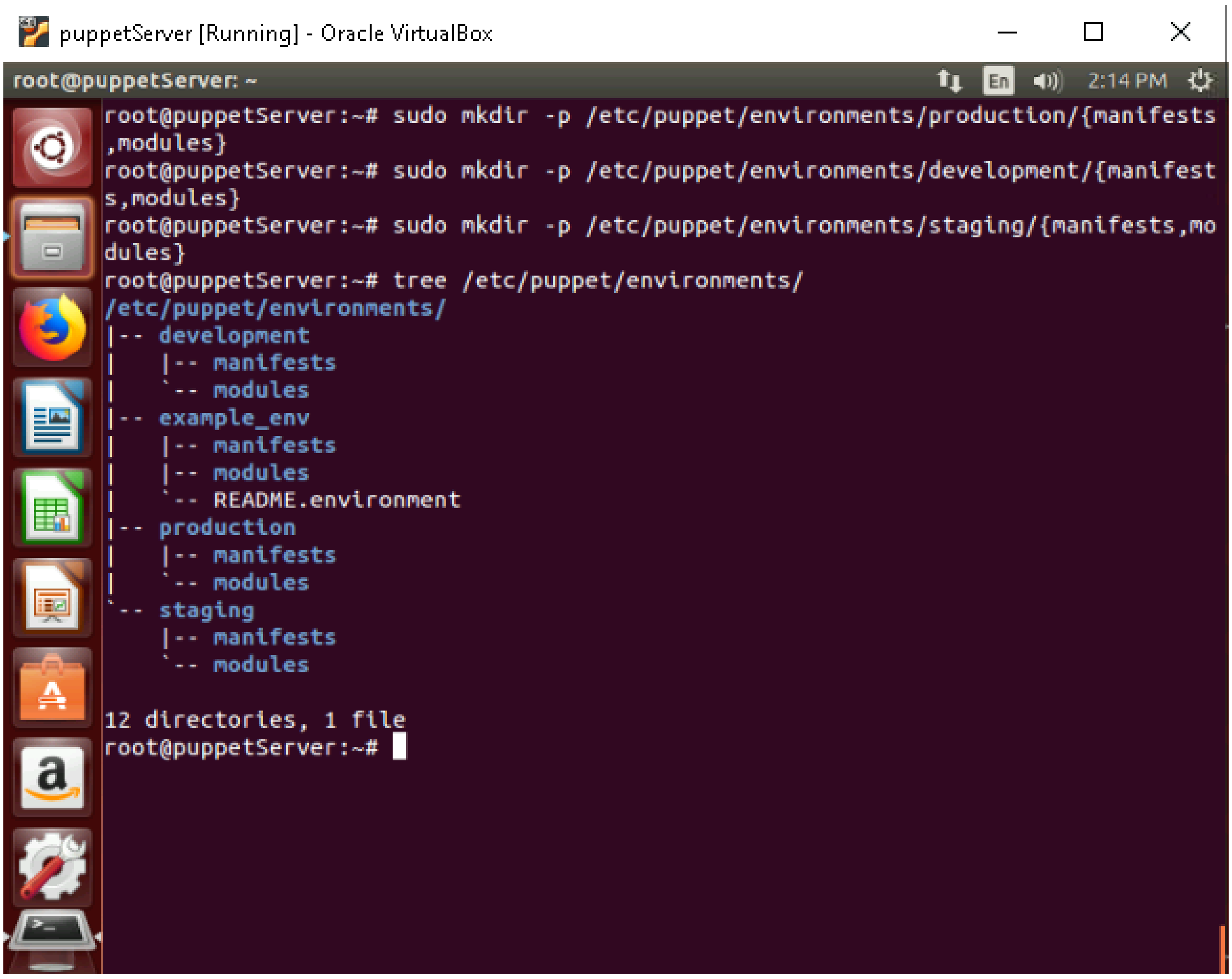
   

   ```
   root@puppetServer:~# sudo mkdir -p /etc/puppet/environments/{production,developme
   nt,staging}
   root@puppetServer:~# tree /etc/puppet/environments/
   /etc/puppet/environments/
   |-- development
   |-- example_env
   |   |-- manifests
   |   |-- modules
   |   `-- README.environment
   |-- production
   `-- staging

   6 directories, 1 file
   root@puppetServer:~#
   ```

2. **Directory Structure**:

   Inside each environment, you can create subdirectories for `modules`, `manifests`, and other necessary files. For example:

```
sudo mkdir -p /etc/puppet/environments/production/{manifests,modules}
sudo mkdir -p /etc/puppet/environments/development/{manifests,modules}
sudo mkdir -p /etc/puppet/environments/staging/{manifests,modules}
```



```
root@puppetServer:~# sudo mkdir -p /etc/puppet/environments/production/{manifests
,modules}
root@puppetServer:~# sudo mkdir -p /etc/puppet/environments/development/{manifest
s,modules}
root@puppetServer:~# sudo mkdir -p /etc/puppet/environments/staging/{manifests,mo
dules}
root@puppetServer:~# tree /etc/puppet/environments/
/etc/puppet/environments/
|-- development
|   |-- manifests
|   `-- modules
|-- example_env
|   |-- manifests
|   |-- modules
|   `-- README.environment
|-- production
|   |-- manifests
|   `-- modules
`-- staging
    |-- manifests
    `-- modules

12 directories, 1 file
root@puppetServer:~#
```

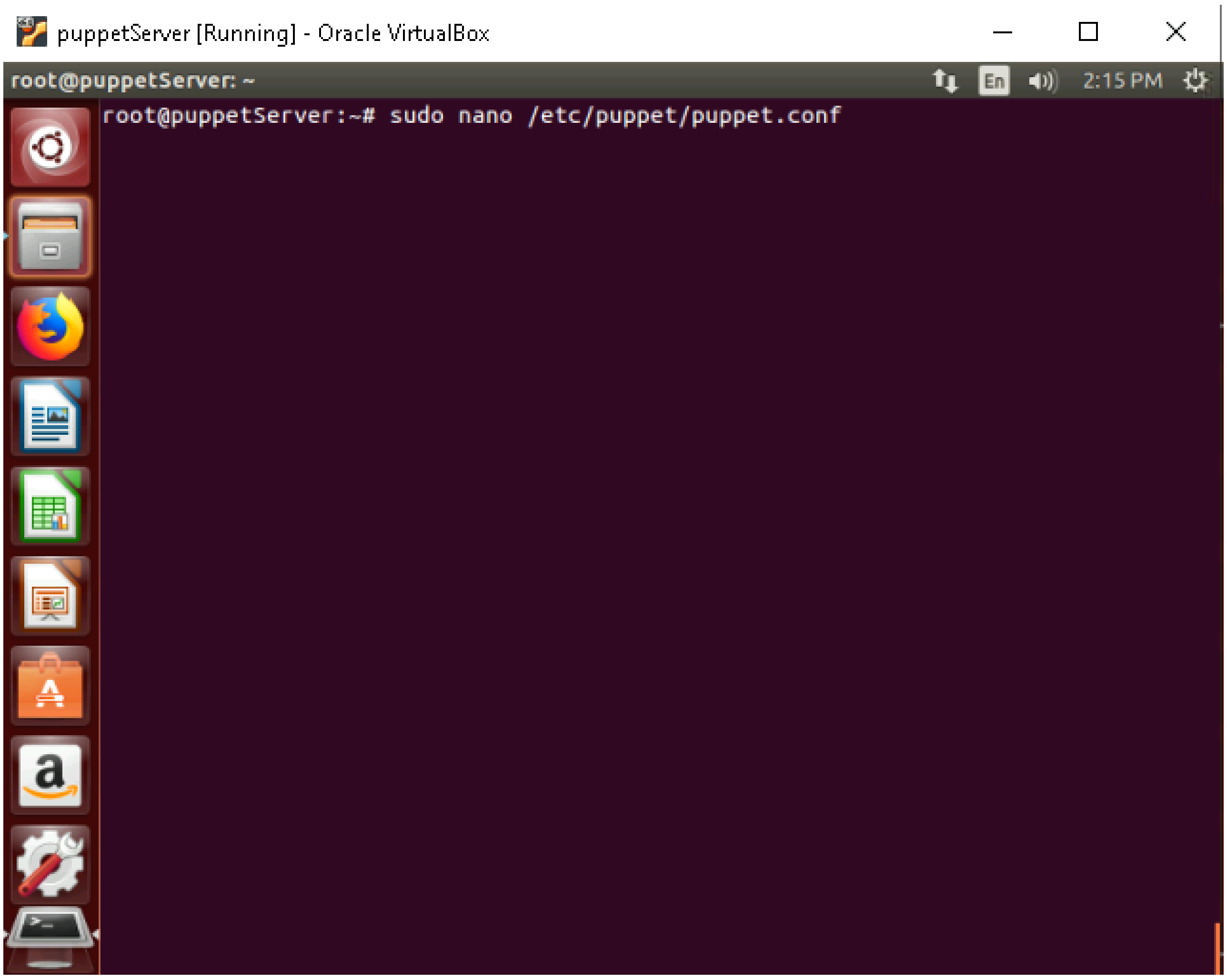
Each environment will have its own set of `manifests` and `modules` that are isolated from one another.

Step 2: Configure Puppet to Use Environments

1. **Edit Puppet Configuration**:

   Open the `puppet.conf` file located at `/etc/puppet/puppet.conf`:

   ```
   sudo nano /etc/puppet/puppet.conf
   ```

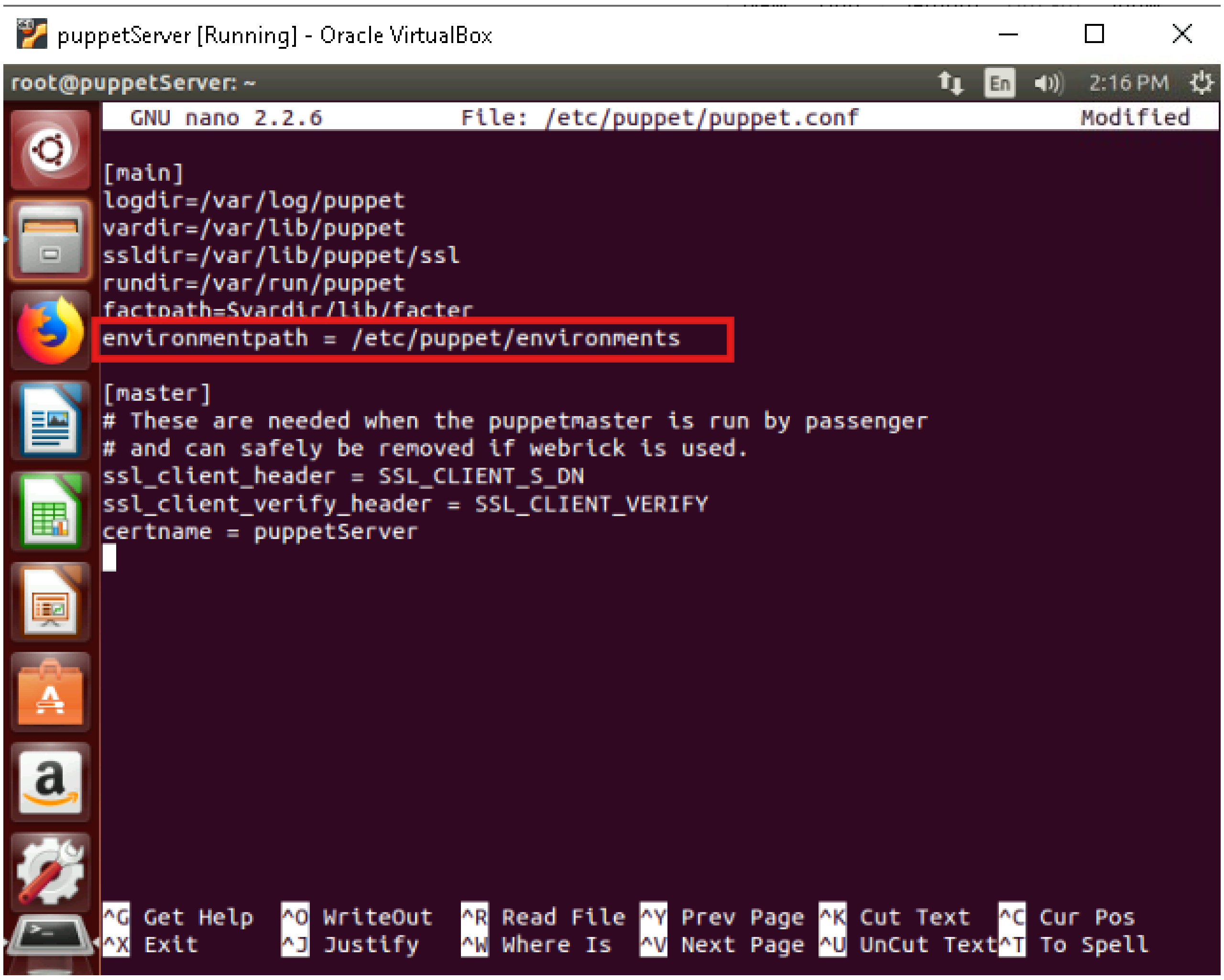


```
root@puppetServer:~# sudo nano /etc/puppet/puppet.conf
```

2. **Set the Environment Path**:

   Add or modify the following lines to enable environment support:

   ```
   [main]
   environmentpath = /etc/puppet/environments
   ```



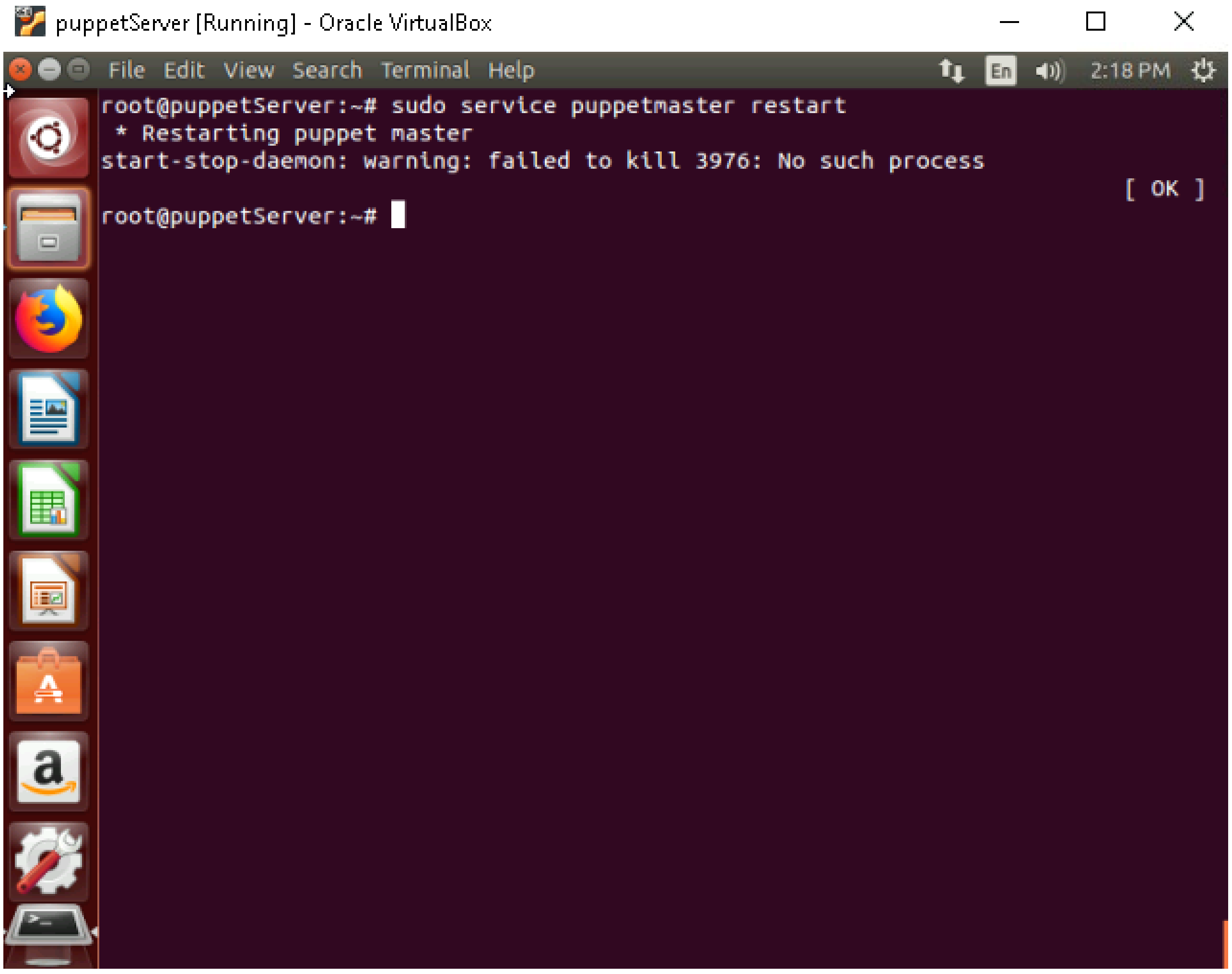
```
  GNU nano 2.2.6           File: /etc/puppet/puppet.conf              Modified

[main]
logdir=/var/log/puppet
vardir=/var/lib/puppet
ssldir=/var/lib/puppet/ssl
rundir=/var/run/puppet
factpath=$vardir/lib/facter
environmentpath = /etc/puppet/environments

[master]
# These are needed when the puppetmaster is run by passenger
# and can safely be removed if webrick is used.
ssl_client_header = SSL_CLIENT_S_DN
ssl_client_verify_header = SSL_CLIENT_VERIFY
certname = puppetServer

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

- **Explanation**: This configuration tells Puppet to look in `/etc/puppet/environments/` for environments.

3. **Restart the Puppet Master**:

   After configuring Puppet to use environments, restart the Puppet Master to apply the changes:

```
sudo service puppetmaster restart
```


puppetServer [Running] - Oracle VirtualBox
File Edit View Search Terminal Help
En
2:18 PM

```
root@puppetServer:~# sudo service puppetmaster restart
 * Restarting puppet master
start-stop-daemon: warning: failed to kill 3976: No such process
                                                                  [ OK ]
root@puppetServer:~#
```

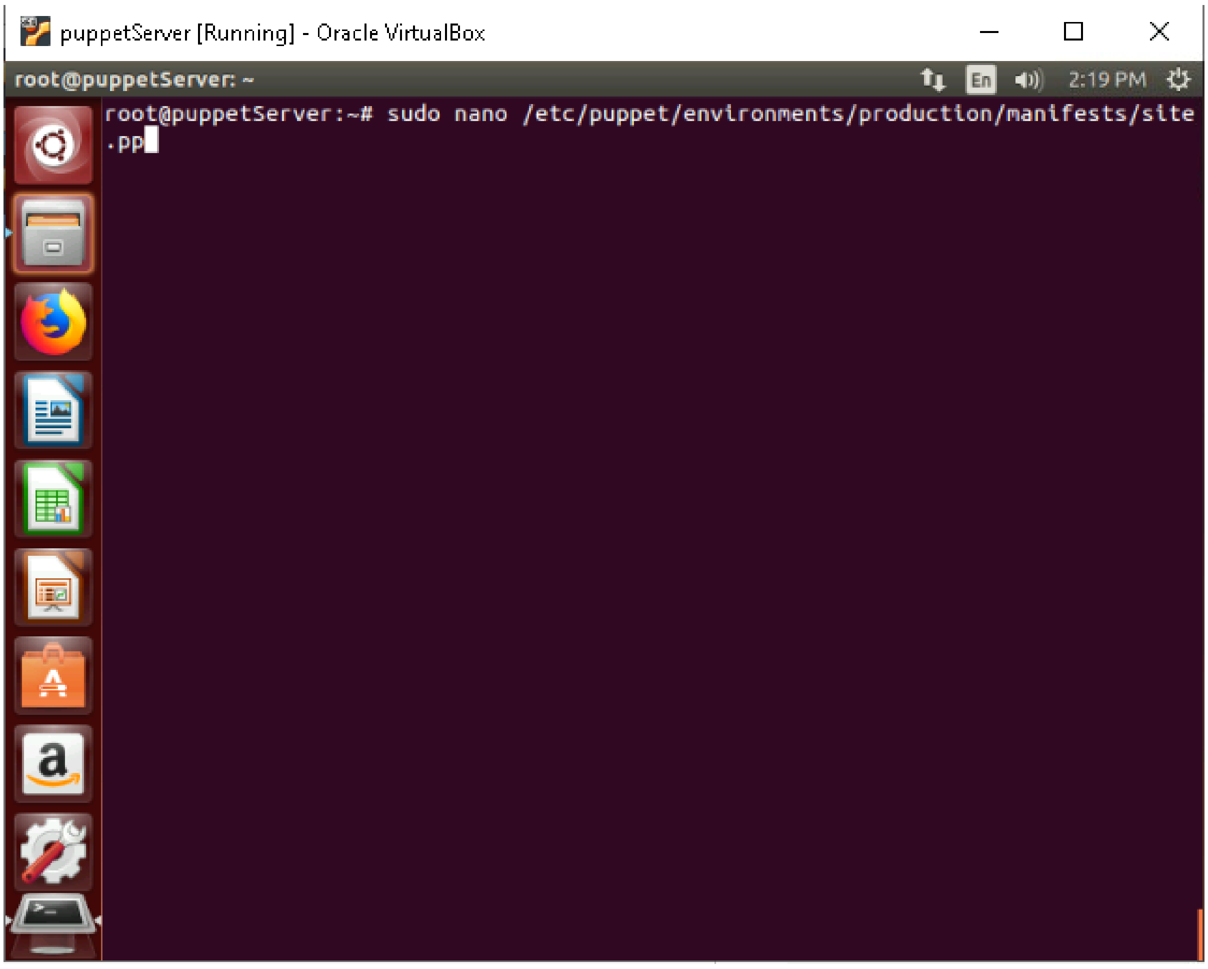
Step 3: Create Environment-Specific Manifests

1. **Create an Environment-Specific Manifest**:

   Each environment will have its own manifest. For example, create a `site.pp` file for each environment:

   - **For the `production` environment**:

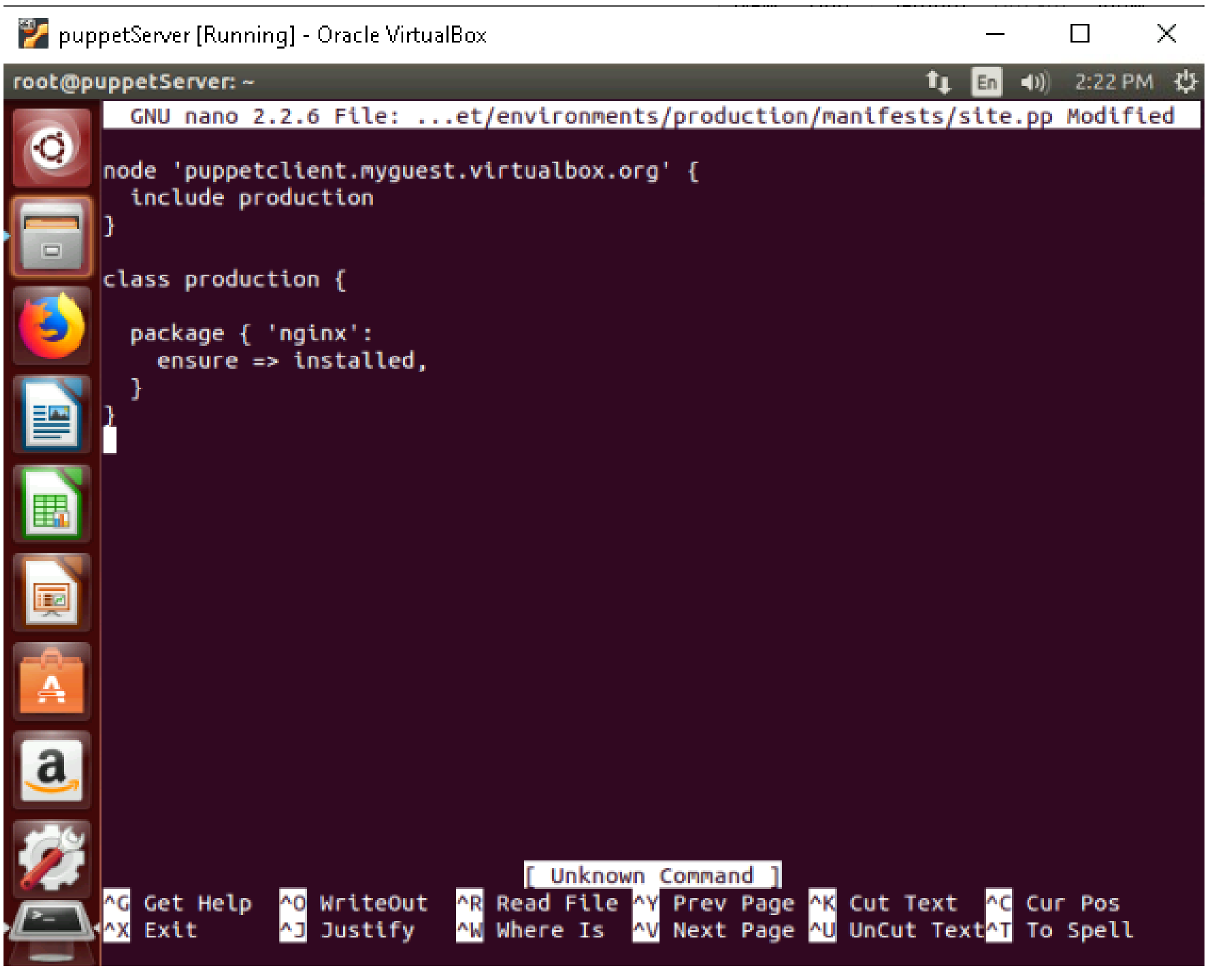
     ```
     sudo nano /etc/puppet/environments/production/manifests/site.pp
     ```



```
root@puppetServer:~# sudo nano /etc/puppet/environments/production/manifests/site
.pp
```

Add configurations specific to the production environment:

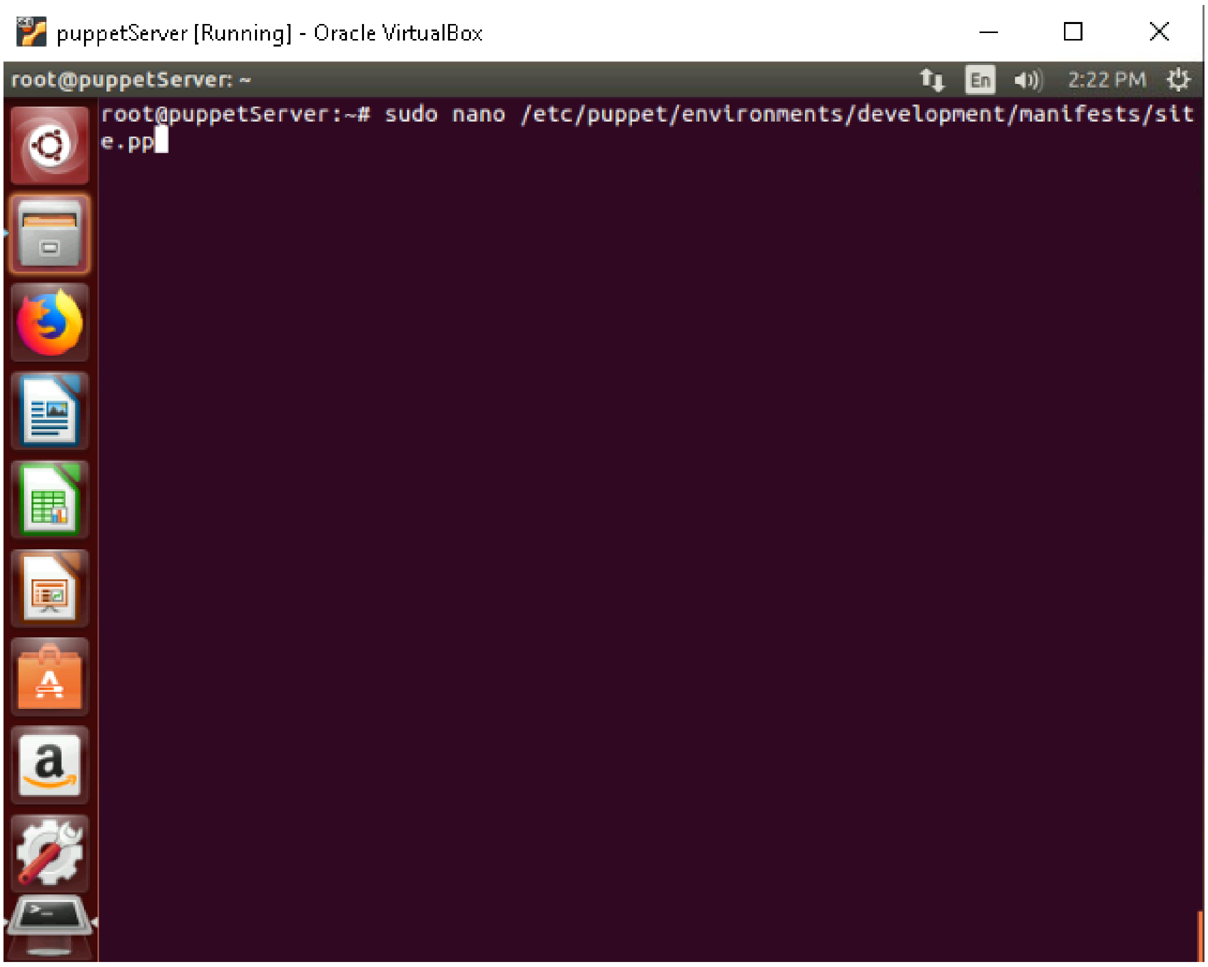
```
node 'puppetclient.myguest.virtualbox.org' {
  include production
}

class production {
  # Define production-specific configurations
  package { 'nginx':
    ensure => installed,
  }
}
```



```
  GNU nano 2.2.6 File: ...et/environments/production/manifests/site.pp Modified

node 'puppetclient.myguest.virtualbox.org' {
  include production
}

class production {

  package { 'nginx':
    ensure => installed,
  }
}

                              [ Unknown Command ]
^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text^T To Spell
```

- **For the `development` environment**:

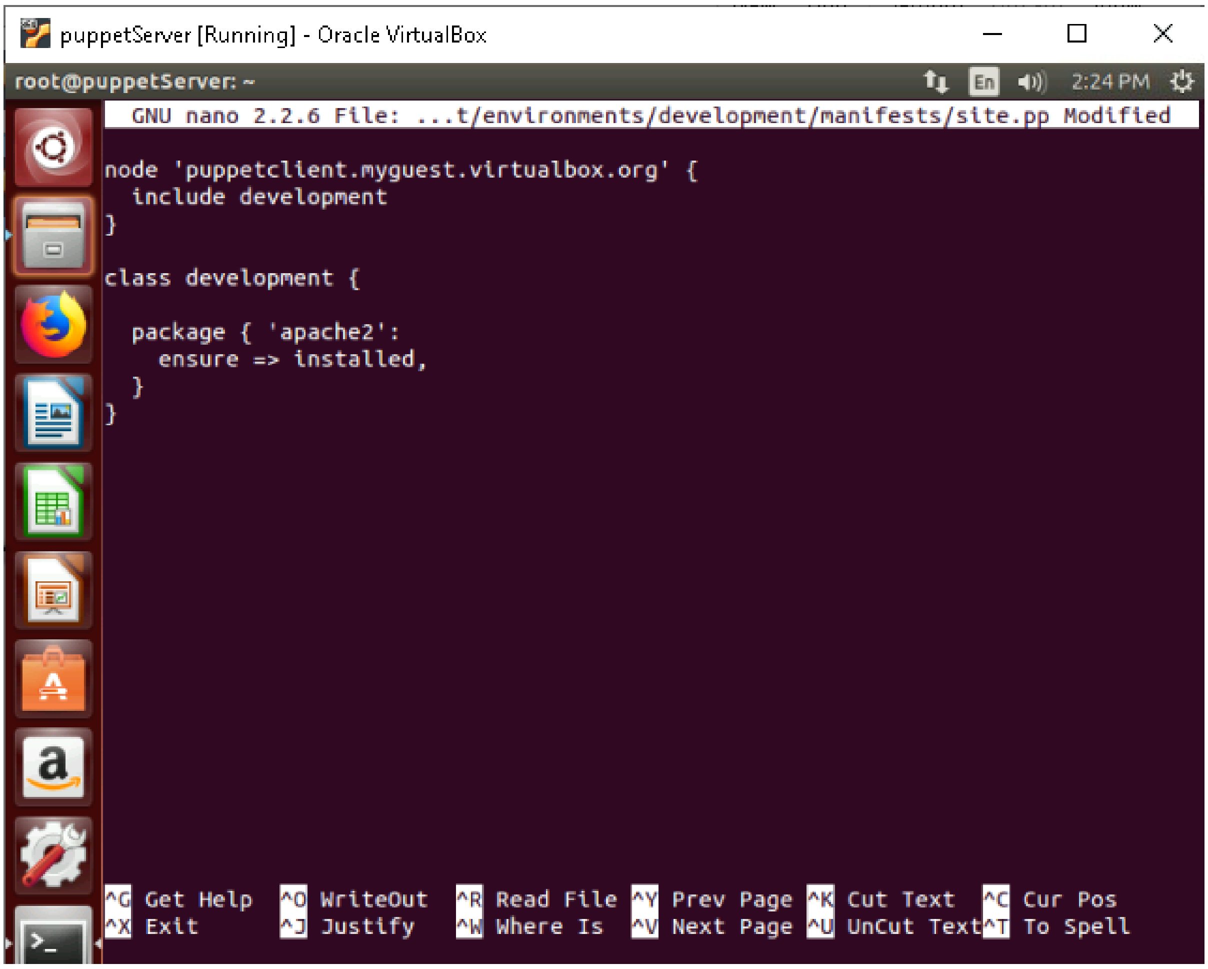
```
sudo nano /etc/puppet/environments/development/manifests/site.pp
```



```
root@puppetServer:~# sudo nano /etc/puppet/environments/development/manifests/site.pp
```

Add configurations specific to the development environment:

```
node default {
  include development
}

class development {
  # Define development-specific configurations
  package { 'apache2':
    ensure => installed,
  }
}
```



```
  GNU nano 2.2.6 File: ...t/environments/development/manifests/site.pp Modified

node 'puppetclient.myguest.virtualbox.org' {
  include development
}

class development {

  package { 'apache2':
    ensure => installed,
  }
}

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

2. **Create Additional Modules (Optional)**:

   You can also create environment-specific modules within the `modules` directory inside each environment:

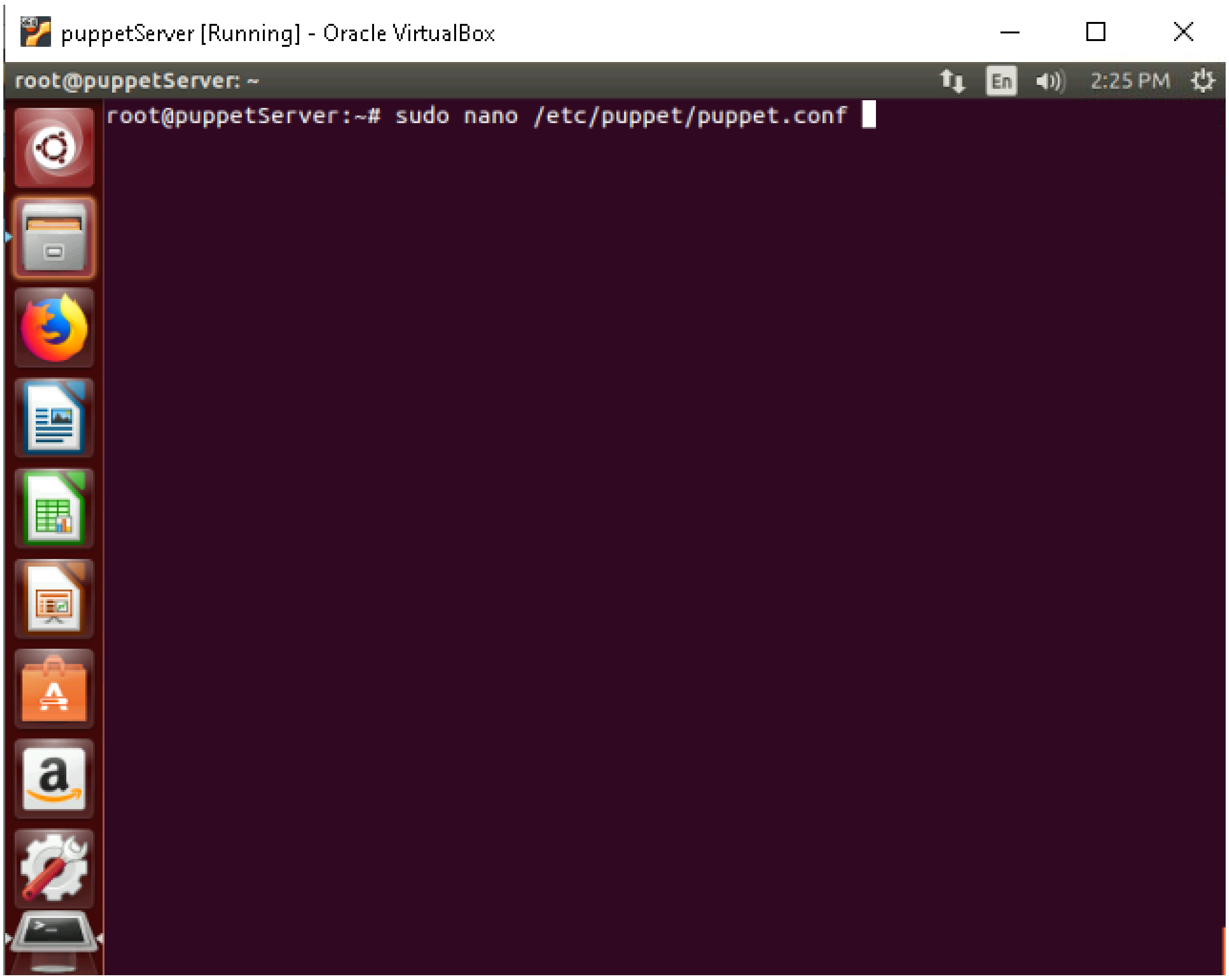
   ```
   sudo mkdir -p /etc/puppet/environments/production/modules/nginx
   sudo mkdir -p /etc/puppet/environments/development/modules/apache
   ```

Step 4: Apply Environments to Nodes

1. **Assign Environments to Nodes**:

   Assign specific environments to Puppet Agent nodes using the `puppet.conf` file or through external node classification (ENC). For example, modify the `puppet.conf` on the agent:

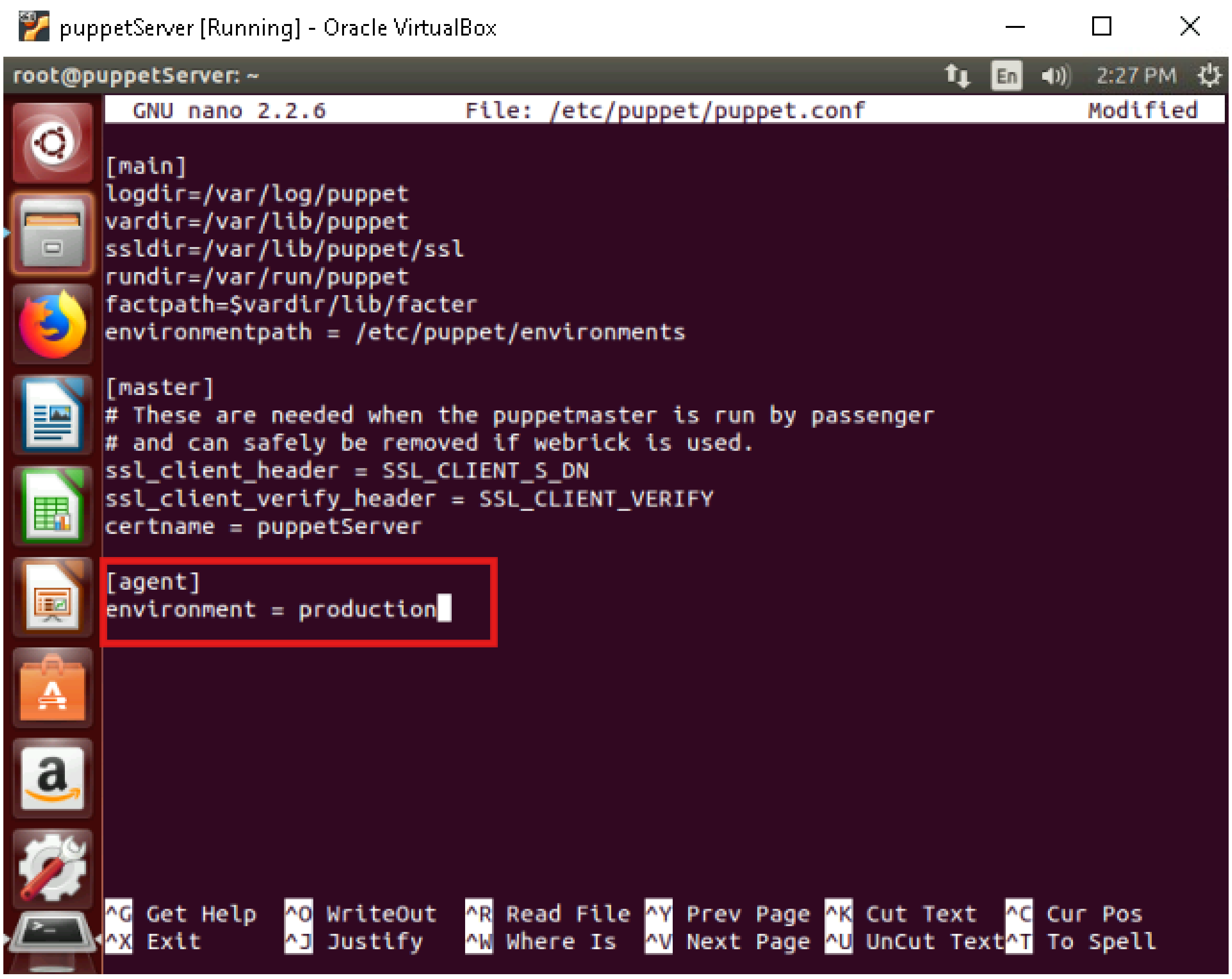
   ```
   sudo nano /etc/puppet/puppet.conf
   ```



```
root@puppetServer:~# sudo nano /etc/puppet/puppet.conf
```

Add the following lines to assign an environment to the agent:

```
[agent]
environment = production
```



```
  GNU nano 2.2.6          File: /etc/puppet/puppet.conf                 Modified

[main]
logdir=/var/log/puppet
vardir=/var/lib/puppet
ssldir=/var/lib/puppet/ssl
rundir=/var/run/puppet
factpath=$vardir/lib/facter
environmentpath = /etc/puppet/environments

[master]
# These are needed when the puppetmaster is run by passenger
# and can safely be removed if webrick is used.
ssl_client_header = SSL_CLIENT_S_DN
ssl_client_verify_header = SSL_CLIENT_VERIFY
certname = puppetServer

[agent]
environment = production

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

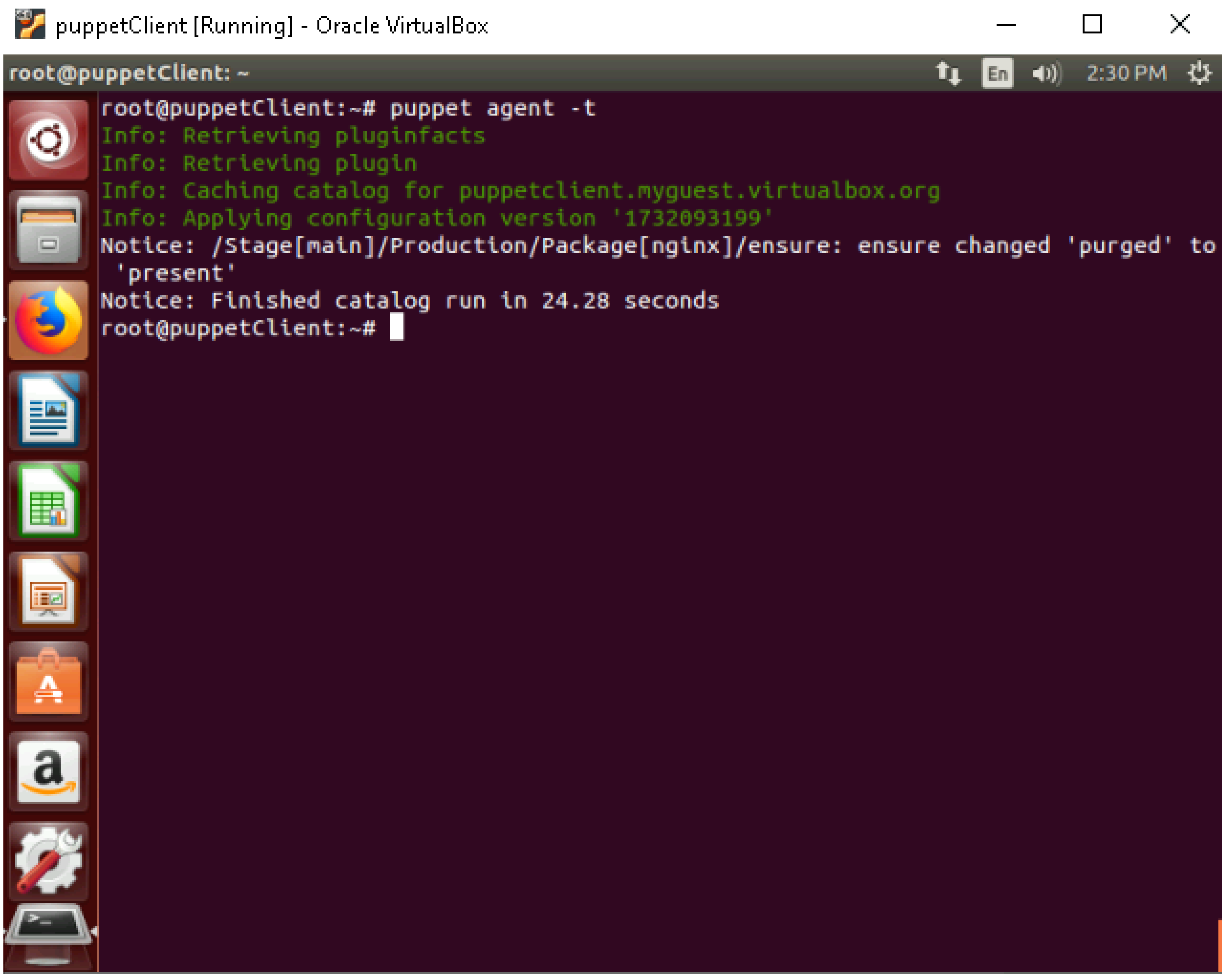
- Note: Make sure to restart the Puppet Master after making changes to the `puppet.conf` file.
- Alternatively, you can use the `site.pp` file on the Puppet Master to assign environments:

  ```
  node 'puppetclient.example.com' {
    environment => 'development'
  }
  ```

2. **Run Puppet on the Agent**:

   Apply the environment configuration on the Puppet Agent:

   ```
   puppet agent -t
   ```



```
root@puppetClient:~# puppet agent -t
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for puppetclient.myguest.virtualbox.org
Info: Applying configuration version '1732093199'
Notice: /Stage[main]/Production/Package[nginx]/ensure: ensure changed 'purged' to
 'present'
Notice: Finished catalog run in 24.28 seconds
root@puppetClient:~#
```

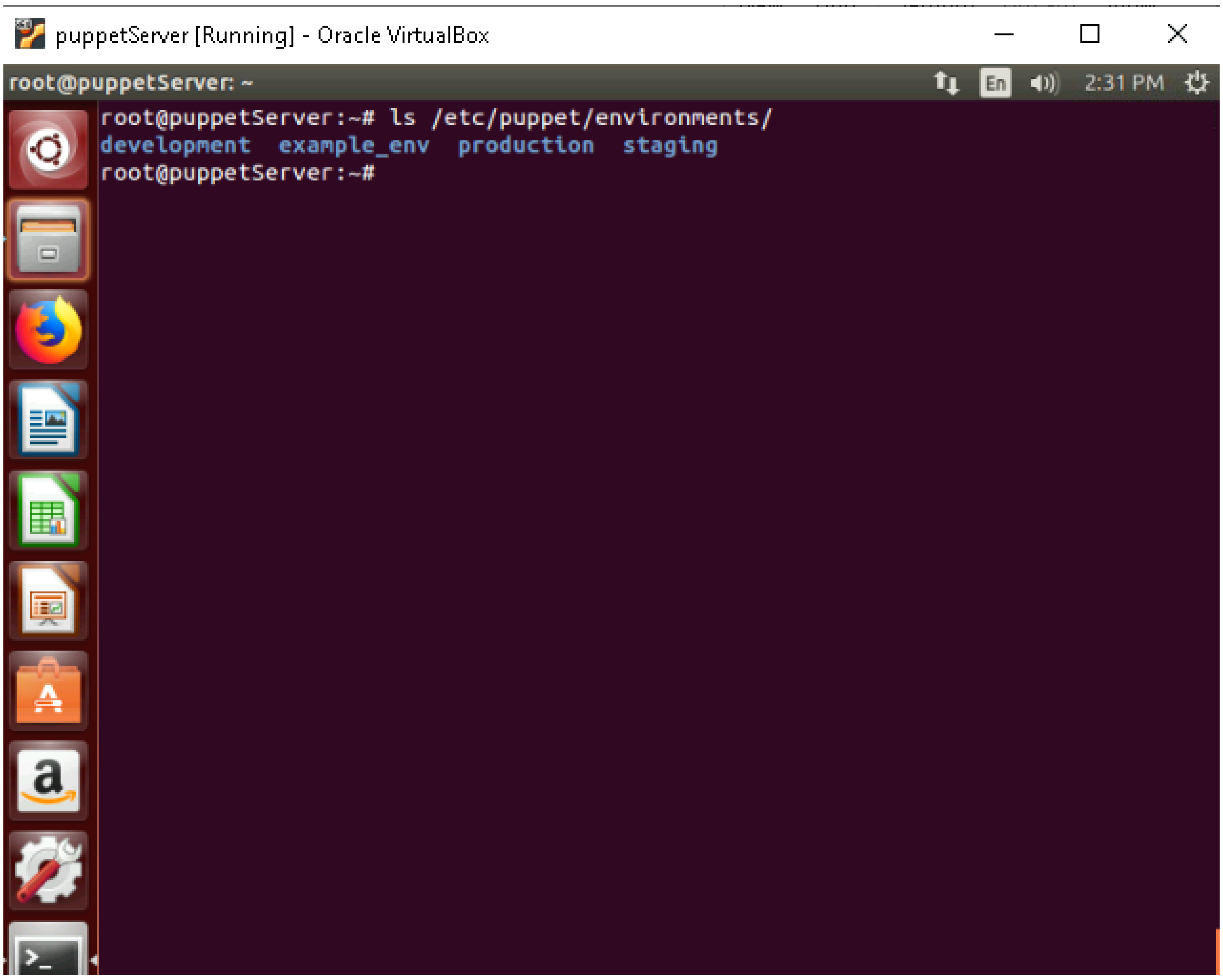
Puppet will fetch and apply configurations from the appropriate environment (e.g., `production`, `development`).

Step 5: Manage Environments in Puppet

1. **List Available Environments**:

   You can list available environments on the Puppet Master by checking the `environmentpath` directory:

   ```
   ls /etc/puppet/environments/
   ```



```
root@puppetServer:~# ls /etc/puppet/environments/
development  example_env  production  staging
root@puppetServer:~#
```

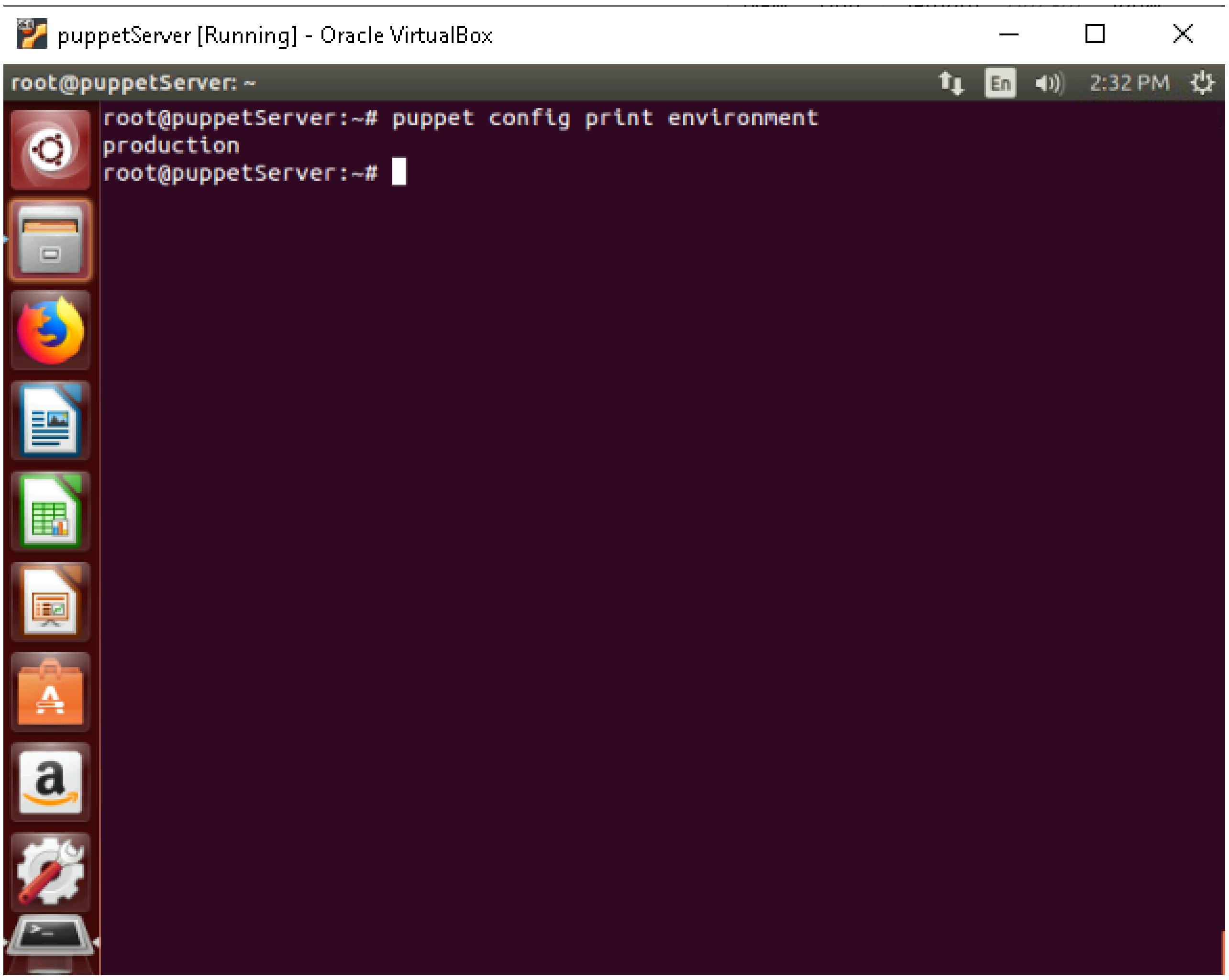
2. **Switch Environments Dynamically**:

   If you need to switch an environment for an agent dynamically, you can modify the node definition in the Puppet Master or use an ENC tool to assign the new environment.

3. **Testing Environments**:

   Verify the environment settings by checking which environment is being applied:

   ```
   puppet config print environment
   ```



```
root@puppetServer:~# puppet config print environment
production
root@puppetServer:~#
```

References

- Puppet 3.8 Documentation: Puppet Docs
- Puppet with Multiple Environments: Puppet Environments Best Practices

---