Puppet is a cross-platform configuration management tool for automating all your IT configurations.

This book kicks off with the Puppet fundamentals and an introduction to the Foreman interface for managing Puppet nodes. Next, you deal with Puppet agents' installation on multiple clients and grouping them to connect to the Puppet Server for easy management. You then learn about Puppet's module structure to familiarize yourself with the Puppet manifests' scripting. You then move on to Puppet Forge modules installation and usage in Windows. You will also venture into the Windows security aspects by gaining insights into various security settings, further securing your server and clients. You will learn how to check the reporting and status monitoring details.

Finally, you will install and update software on multiple Windows clients using Puppet and Chocolatey, thereby taming Puppet for your future projects.

Who this book is written for
This book is for Windows administrators seeking to automate the management tasks of Windows servers and clients. Prior experience in Windows administration and a basic knowledge of Linux and Puppet will help you create wonders.

What you will learn from this book
- Install Puppet on a Windows Server and Foreman interface to have a user-friendly Puppet working environment
- Discover how to install Puppet agents to single or multiple computers and get your hosts connected to Puppetmaster in one shot
- Understand all about creating modules, writing manifests, managing files, services, and users in Puppet
- Learn how to install the Puppet Forge modules to complete tasks such as registry management, ACL, and firewall
- Explore the usage of Puppet facts, functions, and modules to have a deeper understanding of how Puppet functions
- Manage the installation and updates of third-party software using Puppet with Chocolatey to completely forgo the manual update chores in your daily plan
- Use Puppet for Windows security to have a more secure environment

Organize your Windows environment using Puppet tools to unload administrative burdens in a short time.
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Installing Puppet Server and Foreman'
- A synopsis of the book’s content
- More information on Learning Puppet for Windows Server
Fuat Ulugay is currently the IT and ERP director for SOCAR Turkey. He lives in Istanbul, Turkey. Also, he has worked as an SAP ABAP and SAP CRM consultant for more than 10 years.

He is a great fan of open source projects. He implements and teaches them whenever possible. He is good at penetration testing, network security monitoring, industrial control systems security, system administration, and virtualization. Also, he is leading and teaching the security team at his company. He has a blog at http://hacktr.org, where he writes on open source and security-related topics.
Preface

Puppet is a configuration management tool. It allows you to automate all your IT configurations by giving you the control of what you do to each node (Puppet agent), and also, focusing on when and how you do the configurations. In this context, Puppet is a cross-platform tool that is widely used for Unix-like and Microsoft Windows systems. However, it has been popularly used for Unix-like systems.

This book provides insights into using Puppet for Windows administration tasks, such as server setup, application updates, and service management.

This book kicks off with the fundamentals of Puppet by helping you with the installation of Puppet on a Windows Server, and progresses with the introduction of the Foreman interface to manage Puppet nodes. Next, you will deal with the installation of Puppet agents on multiple clients and how to connect them to your Puppet server by grouping your nodes for easy management. Then, you will become familiar with the scripting of Puppet manifests along with an understanding of the module structure in Puppet. You will further move on to the installation of the Puppet Forge modules and their usage in Windows along with advanced topics such as facts, functions, and templates.

Moreover, you will venture into the security aspects for Windows by gaining insights into the various security settings that will make your server and clients more secure from hackers that use different attack vectors. You will also use Puppet and Chocolatey to install and update software.

Finally, you will round off by learning how to check the details of reporting and status monitoring along with the automation of installing and updating software for multiple Windows clients, arming you with ample artillery to tame Puppet for your future projects.
Preface

What this book covers

Chapter 1, Installing Puppet Server and Foreman, starts with an introduction to Puppet. It continues with the installation of the operating system of the server. Next, it deals with the installation of Puppet Server and Foreman. Finally, this chapter ends with the security settings of the server.

In this chapter, we start by learning what Puppet is. Then, we continue with the differences of Puppet implementation. We get hands-on experience by installing Puppet Server and Foreman. The last step is to learn how to keep your server secure. In the next chapter, we will deal with the Puppet agents and their installation on the hosts.

Chapter 2, Installing Puppet Agents, starts with setting up agents for single hosts. Next, it continues with modifying the MSI package for Puppet agent installation. After modifying the MSI file, this chapter shows how to use it to install the Puppet agents on multiple hosts by a third-party software and domain group policy. Lastly, the chapter finishes with the management of host certificates and host groups.

Chapter 3, Your First Modules, starts with writing your first module and continues with some basic module examples for file, directory, service, and user management. It also shows how to import the classes to Foreman and assign them to the hosts or host groups.

Chapter 4, Puppet Forge Modules for Windows, takes us into the world of Puppet Forge, where you can find many ready-to-use modules for Windows. The modules that are explained are registry, ACL, firewall, and reboot.

Chapter 5, Puppet Facts, Functions, and Templates, explains how to write Puppet facts, functions, and templates. This chapter shows how to display facts and write your custom facts. Also, it explains the templates to create dynamic content files. It gives details of the stdlib functions and how to create a custom function.

Chapter 6, Using Puppet for Windows Security, shows practices to make Windows more secure using Puppet. The purpose of this chapter is to make hacking activity harder for hackers and keep our systems as secure as possible. The sample practices are locking the startup folder and hosts file, starting the necessary services and stopping the unnecessary ones, setting the firewall rules, and finally, making the local administrator passwords unique.

Chapter 7, Reporting and Monitoring, shows many details about monitoring and checking the statuses of the host, such as how to see the statuses of the hosts in a summary, what information is available for the hosts, reporting the details of Foreman, and checking the definitions, statuses, and facts of the hosts in the terminal. Finally, this chapter deals with how to see the access and error logs for Foreman and Puppet.
Chapter 8, *Installing Software and Updates*, shows how to install a software using the Puppet package resource. Next, we continue with the details and usage of Chocolatey. Later, we use Puppet and Chocolatey together to make our installations and updates much easier. This chapter checks some of the commonly used software and how to always keep them updated. Finally, this chapter shows how to update the Puppet agents and uninstall the software.
Installing Puppet Server and Foreman

Puppet is a configuration management software that allows the defining and enforcing of the desired state across your nodes and keeps them in this desired state. Nodes are the clients and servers that are connected to and managed by Puppet. Puppet supports both Linux and Windows environments. Also, it is available both commercially and in open source. In this book, we will deal only with the open source version.

However, you may also want to use or give Puppet Enterprise a go, which is the commercial solution. There is a virtual appliance available to download on the Puppet website https://puppetlabs.com/download-learning-vm. You can download and test it. It supports up to ten nodes for free. Of course, using the commercial version is easier as it has complete support and additional functionalities. The problem with it is that, you need to pay for it and have a budget. Following are the additional advantages, if you want to check the Enterprise Puppet. For more details, please check out the URL https://puppetlabs.com/puppet/commercial.

- Event inspection
- Role-based access control
- Puppet Server reporting
- Puppet Enterprise installer
- Puppet Enterprise console
- Puppet Node Manager
While reading books based on Puppet, we come across one little problem: the books are all about managing Linux systems. There are almost no resources explaining the details about managing Windows servers and clients. Thus, it was a challenge for me to use Puppet in the Windows environment. While doing this, I had to deal with many problems and learn the hard way. In this book, I will make it much easier for you to use Puppet for Windows. By the end of this book, you will have a solid understanding of how to write manifests for Windows and deal with the configuration problems. There will be practical step-by-step examples to complete the tasks. However, we will not delve much into technical and theoretical discussions. The book will show you one easy way of doing it. However, this does not mean that this is the only way to do it.

For example, we will use the Foreman web interface with Apache to manage hosts. This does not mean that this is the only way. You can use Puppet without any server and you can distribute the manifests with Git. This is called the masterless mode. You can only implement this with Apache and handle all the manifests from the terminal. You can also use the enterprise version. These are the perfectly possible ways of using Puppet, but may require more effort or money. The list of ways mentioned here is not exhaustive and every implementation method has not been covered. I have tried many ways and different usages, and came to the conclusion that using Foreman with Puppet is one of the easiest ways to start. This maximizes the benefits and minimizes the effort. However, this is subjective and some people may feel more comfortable without the graphical interface, or may switch to the enterprise version.

The differences between using Puppet with Windows and with Linux

Here, I will tell you some basic differences and not deal with an exhaustive list of all the differences between Windows and Linux. When checking out Puppet and writing the manifests, you may realize that it is much easier with Linux but harder to complete the same tasks with Windows. Here are some examples:

- **File resource**: This manages the permissions, ownership, and contents of the files. Permission settings do not work as successfully for Windows, as it works for Linux; we will use ACL module for this purpose.

- **Package resource**: This manages the packages and software installation. For Windows, we cannot directly install a package and keep it updated as we do in Linux, because Window sit does not have a package manager such as aptitude or yum. First, we need to first find the installer and send it to the host to handle the installation.
• **Puppet agent updates**: These are not easy with packages and requires manual steps.

• **Firewall**: This has support for Linux, but not an official support for Windows Firewall. We need to write our own manifests, or we can find a solution from Puppet Forge.

• **Windows Task Scheduler**: This is not fully supported and has only rudimentary functionality.

• **Windows Server**: This has a very limited support.

This list may continue in this way. Thus, as we can see, the differences are not in favor of Windows. In this book, we will solve these types of problems and show you how to handle them in an easy way.

### Installing Puppet Server

We will start with the installation of the operating system of the Puppet Server. From now on, the Puppet Server will be called Puppet Master. We will use the Ubuntu server 14.04 LTS. Some users may prefer Enterprise Linux such as Red Hat or CentOS. If you prefer another flavor of Linux, this is also fine. Following are the server requirements. The requirements are fine for 500 to 1000 nodes. These requirements will change according to the number of your nodes:

- Ubuntu Server 14.04 LTS
- At least 4 GB RAM
- At least 2 Core CPU
- At least 40 GB of hard disk space

You can download the Ubuntu Server 14.04 LTS ISO from [http://www.ubuntu.com/download/server](http://www.ubuntu.com/download/server). Using **Long Term Support (LTS)** versions, ensure that you do not have to upgrade your server for a long time and that there will be few issues about upgrading the distribution. If you want to test it first locally on your computer, you can also download and install VirtualBox from [https://www.virtualbox.org/wiki/Downloads](https://www.virtualbox.org/wiki/Downloads). All the examples in this book have been created in VirtualBox.

You should have a new installation with the OpenSSH server. We will use SSH to connect to the server.
Connecting your server with SSH

We will use SSH to connect to our server. The installation is very easy. You can use the following command to install:

```
$ sudo apt-get install ssh -y
```

- **sudo**: This enables you to run a command with root privileges.
- **apt-get**: The APT package handling utility is used to install and uninstall software.
- **install**: This option is used with apt-get to install a package.
- **ssh**: This is the ssh server package name that will be installed.
- **-y**: The apt-get installation asks, "Do you want to continue [y/n]?". This flag gives the answer as yes and the command runs without interruption.

If you use Windows as your operating system, you can connect using PuTTY. Download this from the link [http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html).

If you use Linux, you can connect from the terminal by the ssh command. For example,

```
$ sshusername@serverip
```

From now on, we will use ssh to connect to our server.

Installing Puppet

Puppet installation usually follows the following steps:

- Set the hostname
- Set FQDN
- Set the static IP, gateway and DNS
- Add the Puppet repositories
- Install Puppet

Let's have a look at each of them.
Chapter 1

Setting the hostname

I will use puppetmaster as the hostname. You can use either vim or nano for text editing. If you have never used vim before, it will be easier for you to use nano.

$ sudonano /etc/hostname

- sudo: For configuration changes, we will need the root privileges. If we do not run the command with sudo, we cannot save our changes to the configuration file.
- nano: This is the command to run the nano text editor.
- /etc/hostname: This is the filename for the hostname configuration.

Use CTRL + X and Y to save.

Setting FQDN

I will use puppetmaster.example.com. Use the following command to edit the /etc/hosts file:

$ sudonano /etc/hosts

Change the contents as follows. Use your own IP according to your network.

127.0.0.1    localhost
127.0.1.1    puppetmaster.example.com
10.10.10.10  puppetmaster.example.com puppetmaster
To verify that the changes are effective, use the `hostname` and `hostname -f` commands.

```bash
puppet@puppetmaster:~$ hostname
puppetmaster
puppet@puppetmaster:~$ hostname -f
puppetmaster.example.com
puppet@puppetmaster:~$
```

You also need to add the IP and **fully qualified domain name (FQDN)** to your company DNS, so that the other computers can find your server. I assume that, as a Windows system administrator, you already know how to do this. The IPs used here may not suit your network and IP ranges, so please change all the IP details throughout the book according to your needs.

### Setting static IP, gateway, and DNS

For your server, give a static IP and define your gateway and nameserver IPs.

First, define your IP gateway and subnet mask. For this, we will edit the `/etc/network/interfaces` file. Following is the sample detail I have added for my Puppet Master:

```bash
$ sudo nano /etc/network/interfaces
```

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
  address 10.10.10.10
  netmask 255.255.255.0
  broadcast 10.10.10.255
  gateway 10.10.10.1
```
As you can see, the details are self-explanatory:

- **eth0**: This is the network interface name
- **address**: This is your server's IP
- **netmask**: This is the subnetmask
- **broadcast**: This is the broadcast IP
- **gateway**: This is the gateway IP

Now, let's set the nameserver IPs. To set NS records, we need to edit `/etc/resolvconf/resolv.conf.d/base`, as follows:

```bash
$ nano /etc/resolvconf/resolv.conf.d/base
```

The sample contents of this file are shown in the following screenshot. In our example, we are using the Google DNS IPs. Here, you can use your company's DNS IPs.

![Sample content of resolv.conf](image)

---

**Downloading the example code**

You can download the example code files from your account at http://www.packtpub.com for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit http://www.packtpub.com/support and register to have the files e-mailed directly to you.
Installing Puppet Server and Foreman

After saving the file, reboot the server. After rebooting, you can check using the following command whether you have the correct IP, netmask, and broadcast:

```bash
$ ifconfig
```

- **ifconfig**: This command is used to get info and make the changes in the network interfaces
- **eth0**: This is the network interface name
- **inet addr:10.10.10.10**: This is the IP address of our server
- **Bcast:10.10.10.255**: This is the broadcast IP
- **Mask:255.255.255.0**: This is the netmask IP

Check whether the gateway is correct using the following command:

```bash
$ route -n
```

- **route**: This command shows and manipulates the IP routing table
- **-n**: This flag is to show the address details in numeric format instead of hostnames
Lastly, check whether nameserver works correctly. Run the `nslookup` command and enter any address, as follows:

$ nslookup

- `nslookup`: This is the command to interactively query the Internet nameservers. If no nameserver is provided, this will use the default one.
- `google.com`: When we enter any hostname, this will provide the details about it
- `exit`: This command ends `nslookup`

As you can see in the preceding screenshot, it checks from 8.8.8.8. Now we can say that our network settings are correct. To exit `nslookup`, you can use the `exit` command.

**Adding the Puppet repositories**

We will first add the Puppet repositories for installation. Secondly, we will update the repositories. Lastly, we will update our server before installing Puppet.

Here are the details to add the Puppet repositories:

$ sudo wget https://apt.puppetlabs.com/puppetlabs-release-trusty.deb
$ sudo dpkg -i puppetlabs-release-trusty.deb

- `wget`: This is a utility for non-interactive downloads of files from the Web. We use this here to download the `puppetlabs-release-trusty.deb` file.
- `dpkg`: This is the package manager for Linux Debian. The option `-i` is used for installation.

Now, update the repository info, as follows:

$ sudo apt-get update
Lastly, install the updates, as follows:

```bash
$ sudo apt-get upgrade -y
```

- **update**: This option is used to resynchronize the package index files from their resources
- **upgrade**: This option is used to upgrade to the newest version of the already installed software

### Installing Puppet

Now, it is time to install Puppet Master and its agent on your server. We will also do some configurations. The command to install Puppet is as follows:

```bash
$ sudo apt-get install -y puppetmaster puppet
```

Now, let's make sure that Puppet Master starts automatically. For this purpose, we need to edit `/etc/default/puppetmaster`. You need to change the `START=no` value to `START=yes`. If it is already `yes`, you can leave this as it is and continue. You also need to enable puppet agent on the server. We will edit the `/etc/default/puppet` file. Again, we need to make sure that `START=yes` exists. We also need to change the server details for the puppet agent. Change the `/etc/puppet/puppet.conf` file and enter the server details. Just after `[main]`, add the line `server=puppetmaster.example.com`.

```
[main]
  server=puppetmaster.example.com
  # The Puppet log directory.
  # The default value is '$vardir/log'.
  logdir = /var/log/puppet

  # Where Puppet PID files are kept.
  # The default value is '$vardir/run'.
  rundir = /var/run/puppet

  # Where SSL certificates are kept.
  # The default value is '$confdir/ssl'.
  ssldir = $vardir/ssl

  # Allow services in the 'puppet' group to access key (Foreman + proxy)
  privatekeydir = $ssldir/private_keys { group = service }
  hostprivkey = $privatekeydir/$certname.pem { mode = 640 }
```

After making the changes, we will restart the services as shown here:

```
$ sudo service puppetmaster restart
$ sudo service puppet restart
```

Check whether they are running using the following commands:

```
$ sudo service puppetmaster status
$ sudo service puppet status
```

As you can see in the preceding screenshot, the services are running without any problem. So we completed the installation of Puppet Master and Puppet agent on our server.

## Installing Foreman

After installing Puppet Master, our next step is to install the Foreman web user interface that will be used to manage and report. Foreman is an open source project that can be used with Puppet or Chef. With Foreman and Puppet, you can manage your servers for configuration management, orchestration, and monitoring.

For installation of Foreman, we will first add the relevant repositories, and after that install it.

Add repository details, as follows:

```
$ sudo -i
# echo "deb http://deb.theforeman.org/ trusty stable" > /etc/apt/sources.list.d/foreman.list
# echo "deb http://deb.theforeman.org/ plugins stable" >> /etc/apt/sources.list.d/foreman.list
```

You can also use `sudo nano` to add the details to the relevant files. In the preceding commands, `echo` outputs the text to the screen or a file. `>` overwrites the file, if it exists; if it does not exist, the file will be created. `>>` adds content to the end of the file without overwriting it.
Installing Puppet Server and Foreman

We need to add the key for the repository, as we are manually adding the source details using this command:

```
# wget -q http://deb.theforeman.org/pubkey.gpg -O- | apt-key add -
```

Now we have successfully added the repository details; we can continue with the repository updates, as follows:

```
# apt-get update
```

We need Apache as our web server. So, we install apache2 and `foreman-installer` as shown in the following:

```
# apt-get install -y foreman-installer
```

The final step is to run the `foreman-installer`. It will take some time to complete. For my server, it took more than five minutes.

```
# foreman-installer
```

Do not forget to write down the user and password details to connect your server. In the next section, we will see the basics of the user interface.
The Foreman interface

Now, just open a browser and enter your URL. In our example, the URL is https://puppetmaster.example.com. You will get the login page.

Log in with your username and password and you will get a screen as follows. In this screen, you will see only one host, which is our Puppet Master server. We added its agent before, and now we can see that it was connected just two minutes ago. This means that we can even manage Puppet Master with Puppet.
The next step is to change your password. From the right upper corner, go over to Admin User and click on My account. Set a new password as per your convenience. Make sure that the password is strong.

Now let's check out the dashboard. For the dashboard, from the Monitor menu click on Dashboard. In the dashboard, you will see the summary statuses of all your hosts. At the moment as we have only one host, there is not much detail. When we will have more hosts, we will check again, and then find that there will be many different statuses. Most of the statuses of your hosts will fall under one of the statuses listed, as follows:

- Hosts that have modifications without any error
- Hosts with errors
Chapter 1

- Hosts with good reports in the last 35 minutes
- Out-of-sync hosts

We will finish reviewing the Foreman interface. This will be enough for now. We will later see more details of the interface to manage our hosts.

Keeping your server secure

Puppet Master must be protected well. It is a high-risk asset. With Puppet, you can install software on all your servers and clients. Puppet agents on Linux and Windows run with root and admin rights respectively. Think about a scenario where a hacker gets control of it. He can easily run any command, install backdoors, and fully compromise your IT infrastructure.

I suggest using the Ubuntu Server version that does not have a graphical user interface. This will ensure that there will not be much unnecessary software on your Puppet Master. Having only the necessary software on it helps you to have a more secure server, and also to keep the performance higher.
Installing Puppet Server and Foreman

Security rule:
Do not install any software that is not necessary. If vulnerable software does not exist, it cannot be attacked.

Backups
Keep frequent backups of your server.

There are many good backup solutions such as Backup Exec and Veeam. There are also hardware level backup solutions for storage devices. As a last option, you can also use free backup solutions such as Burp backup or a version control system (VCS). VCS will only back up your code and configuration.

Backup solutions and how to handle them is out of the scope of the book. Every company and system administrator has or should have their backup solution. If you do not have one, it is really time to implement one of the solutions as soon as possible. Without backups, your most important concern and risk will be business continuity.

Keeping your server up to date
Every day we see that there are new vulnerabilities, and with new security updates they are patched. So you must have a good policy of updates. The updates are very easy with Ubuntu; you only run two commands, as follows:

$ sudo apt-get update
$ sudo apt-get upgrade

I will not explain these commands again. You can check the section Installing Puppet in this chapter to see the details if you need.

Before updating your server, ensure that in case of problems, you can go back. If you are using a virtual machine, such as Puppet Master, take a snapshot before the updates. If you use hardware, take a new backup before you start.

Lastly, it is also a good idea to have a development environment. Testing the updates in development and then updating the production server would be a good practice.
Do not enable root account

Some administrators do not feel happy about entering a `sudo` command and password whenever there is a need for root privilege. To simply bypass this problem, they enable the root account and use it for every task.

The problem with using the root account is that, you have many processes and software running with root privileges. Assuming that one of them is vulnerable, and a hacker targets your server. When the hacker uses this vulnerability and opens a shell, the privileges that he will have are directly related to the process. So, if the process runs with root privileges, the hacker gains root access. Root access implies total control of the server. If the process is using a limited user account, the hacker will gain these privileges. This means that he still has a long way to go and find some way to escalate the privileges.

Also, when you use the root account, you also need to protect the server from yourself. The root account can do everything; with great power comes great responsibility. If you do something wrong, accidentally, you may need to restore your server from a backup. If you do not have a backup, things may get worse.

Always use a limited account and use `sudo` only when needed. This will protect you from hackers and also from yourself.

Check status of the root account, as follows:

```bash
$ sudopasswd root -S
```

If not locked, lock it using the following:

```bash
$ sudopasswd root -l
```

From now on, in this chapter, all the following details are not related to the implementation and installation of Puppet. However, it is suggested to complete these steps in a live environment. When learning in a test environment, security may not be your initial concern. In this case, simply go to Chapter 2, *Installing Puppet Agents*. 
The user password policy

Another important point is to have a decent password policy. With the correct password policy, we will make it harder for the passwords to be cracked. In the Windows Group Policy, there are also settings for password policy. Thus, the Windows administrators can easily understand its necessity. Here are some points to improve upon.

Do not use old passwords that have been used before

To limit the old passwords that can be used, we need to edit the /etc/pam.d/common-password file. PAM is Pluggable Authentication Modules. PAM enables us to change the authentication process of Linux.

```
$ sudo nano /etc/pam.d/common-password
password [success=1 default=ignore] pam_unix.so obscure sha512 remember=5
```

- **pam_unix.so** is the default PAM module
- **obscure sha512** will encrypt the new passwords with sha512
- **success=1** skips the next rule
- **remember=5** will prevent the user from using the last five passwords

Using at least a 10 char complex password

To set more complex passwords, we will install the libpam-cracklib library, as follows:

```
$ sudo apt-get install libpam-cracklib
```

After this, we again edit the /etc/pam.d/common-password file as follows:

```
$ sudo nano /etc/pam.d/common-password
password requisite pam_cracklib.so retry=3 minlen=10 difok=3 ucredit=-1 lcredit=-1 dcredit=-1 ocredit=-1
```

- **retry=3** ensures that while setting the password, if the user cannot successfully set a password three times, the passwd command will abort
- **minlen=10** is the minimum length for the password
• difok=3 is the minimum number of characters that must be different from those of the previous password
• ucredit=-1 sets the minimum number of required uppercase characters to 1
• lcredit=-1 sets the minimum number of required lowercase characters to 1
• dcredit=-1 sets the minimum number of required digits to 1
• ocredit=-1 sets the minimum number of required symbols to 1

Here are the /etc/pam.d/common-password details after the changes:

```bash
# here are the per-package modules (the "Primary" block)
password        requisite    pam_cracklib.so retry=3 minlen=10 difok=3 ucredit=-1 lcredit=-1 dcredit=-1 ocredit=-1
password        [success=1 default=ignore]    pam_unix.so obscure use_authtok try_first_pass sha512 remember=9
password        requisite    pam_deny.so
# here's the fallback if no module succeeds
password        required    pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
password        required    pam_deny.so
# and here are more per-package modules (the "Additional" block)
# end of pam-auth-update config
```

### Expiring password in 90 days

The password expiration details are in the /etc/login.defs file. Change the value of PASS_MAX_DAYS to 90 and it will be forced to update the password every 90 days, as follows:

```
$ sudo nano /etc/login.defs
PASS_MAX_DAYS 90
```

### Locking account

This policy makes sure that any brute force attempt will fail, or need too much time to complete. We will lock the user account for 10 minutes, if five times there are consecutive login failures. For this purpose, we need to modify the /etc/pam.d/common-auth file. After the lock and wait time, if the user successfully logs in with the correct password, the failed attempts counter will be reset to zero. Otherwise, each failed attempt after the lock will cause another 10 minutes of lock. So, I suggest here that you keep a backup user with the sudo rights, as follows:

```
$ sudo nano /etc/pam.d/common-auth
auth    required    pam_tally2.so deny=5 onerr=fail unlock_time=600
```

- pam_tally2.so is the PAM module that comes with Ubuntu installation and used for account locks.
- deny=5 is the setting used to set the number of failed logins to lock an account.
Installing Puppet Server and Foreman

- `onerr=fail` if something weird happens, the PAM login status will be fail or success. The default status is fail.
- `unlock_time=600` is the number of seconds after which the account will be unlocked.

After completing all the changes, restart your server with the `reboot` command for all the changes to be applied. This is done as follows:

\$ sudo reboot

You can check a user's login attempts with the following command:

\$ sudo pam_tally2 -u username

- `pam_tally2` is the command for the login counter PAM
- `-u` is used to set the user

Here is an example output after two unsuccessful login attempts:

```
root@puppetmaster:~# pam_tally2 -u puppet2
Login: Failures: Latest failure: From:
puppet2: 2: 02/25/15 10:10:29: 10.10.10.2

root@puppetmaster:~#  
```

If you want to reset the counter and unlock the user, you can use the following command. `-r` switch is used to reset the failures counter.

\$ sudo pam_tally2 -u username -r

**Using SSH with key file to connect**

This step makes sure that even if somebody gets your password, it will not be usable. Thus, we will use the SSH key files. The SSH key files are used to identify yourself to an SSH server using the public-key cryptography and challenge-response authentication. We will disable the password logon option and it will be only possible to connect with a key file. We will also put a password to the key file, to make sure that it is also not usable without the password.
Creating the public and private key
Use the `ssh-keygen` command to generate the keys, as follows:

```
$ ssh-keygen
```

```
puppet@puppetmaster:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/puppet/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/puppet/.ssh/id_rsa.
Your public key has been saved in /home/puppet/.ssh/id_rsa.pub.
The key fingerprint is:
b1:77:55:2a:00:9b:19:4f:cc:c7:c2:6a:ee:c1:c5:55 puppet@puppetmaster
The key's randomart image is:
+++-[ RSA 2048]-----
 | 0++ .E .|
 | 0=+ 0 |
 | =00+ 0 |
 | 000 0 |
 | +S.. |
 | ++. |
 | .. |
| . |
| +------------------ |
```

`ssh-keygen` first asks for the folder to save the keys. Just push the `Enter` key and continue. The next question is the passphrase. Make sure that you enter a password. Using the password with your key makes sure that, when somebody gets your key, it will be unusable.

Now go to the `.ssh` folder, as follows:

```
$ cd .ssh
```

Under this folder, you will see these two files:

- `id_rsa`: Private key
- `id_rsa.pub`: Public key
Installing Puppet Server and Foreman

We need to add the details of the public key to a file called `authorized_keys`. This will make it possible to log in with the private key.

```sh
$ cat id_rsa.pub > authorized_keys
```

- The `cat` command displays the content of a file in the terminal
- The `>` symbol adds the content of a command output to a file and if the file has content, it will be overwritten
- The `authorized_keys` is the file that will be required for the `ssh` connection with a key

Here is an example screenshot:

```
puppet@puppetmaster:~$ cd .ssh
puppet@puppetmaster:~/.ssh ls
id_rsa  id_rsa.pub
puppet@puppetmaster:~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABGQEAAYpFm3ULyOLGXYHMSU5A584F4DJlnitonB
MBEjw5/lM6paZAbDl1Gsy8hB2O+4oZ9Mk0ERtzmzAQIyVM/bnRRAxSKYToG6pRjMGb/J7jx7+yXKThEeooaqh7W5gk7
GprxN9ba1JpOBlk5Yv3Izicr9B0C8Q9D0b9Ij39IeYkD9v/ygQDI7531cAFGxseE55nQjJx+JWF7JHNW30D0OOkP0b9j33
cOoab6RPGVEn6zwMjiuOBQ1vPz7AR.160gKu7dE9/TwTJ puppet@puppetmaster
puppet@puppetmaster:~/.ssh id_rsa.pub > authorized_keys
puppet@puppetmaster:~/.ssh ls
```

Change the `authorized_keys` file permissions, otherwise the key will not work. This is done as follows:

```
$ chmod 600 authorized_keys
```

This command will allow only the user to edit and write the file; any group or other users will not be able to change it.

**Getting the key to your computer and converting it into the PuTTY format**

Now, we need to get the contents of `id_rsa` to our computer that we will use to connect to Puppet Master.
To do this, you can use FileZilla to connect and download the file. Or you can use any FTP client with SFTP support. Here is a screenshot for the SFTP connection definition:

![Site Manager screenshot](image)

You need to enter the following:

- **Host**: The server IP is **10.10.10.10**. This is the IP that we gave to our Puppet Master server while installing.
- **Protocol**: SFTP.
- **Logon Type**: Normal.
- **User**: Your username.
- **Password**: Your password.
Installing Puppet Server and Foreman

After you have filled the details, click the **Connect** button. When you connect, you will see your user folder and the .ssh folder, as follows:
Download the `id_rsa` file to your documents folder. After this, you need to convert the file with `PuTTYgen.exe`. Open `PuTTYgen.exe` and click the **Load** button. You also need to select the **All Files (*.*)** option, as shown here:

![PuTTYgen Interface](image)

After you click it open, it will ask for the password. After this, click on **Save private key** and give a name to your key. I saved it as `puppetmaster.ppk`. Now, we are ready to use this key to connect to Puppet Master.
Installing Puppet Server and Foreman

To do it, first fill in the details as follows:

- **Host Name**: `username@ipaddress`
- **Port**: 22
- **Saved Sessions**: Give a name for the session
Now go to SSH | Auth. Select the key file, as follows:

![PuTTY Configuration](image)

After this, go back to Session from the Category section and use the Save button. This will make sure that you can reuse the connection and do not have to define the same settings again.

Now you will be able to connect by double-clicking your saved session name. It will ask for the key file password and then you will be able to connect.

**Connecting from Linux**

To connect from Linux systems, you do not need to convert the private key. You can just connect from the terminal with the ssh command, as follows:

```
$ sshusername@serverip -i keyfile
```

When I change this command to my example, it will be as shown here:

```
$ sshpuppet@10.10.10.10 -i id_rsa
```
Disabling the SSH logins with a password
This is our last step for SSH. After disabling the password login option, there will be no possibility for a hacker to use brute force against an account. Also, there will be no possibility to log in, even if the hacker knows the password.

We need to change the /etc/ssh/sshd_config file to disable password, as follows:

```
$ sudo nano /etc/ssh/sshd_config
```

Find the `PasswordAuthentication` text, uncomment it, and set the value to `no`.

```plaintext
# To enable empty passwords, change to yes (NOT RECOMMENDED)
PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with # some PAM modules and threads)
ChallengeResponseAuthentication no

# Change to no to disable tunnelled clear text passwords
PasswordAuthentication no

# Kerberos options
```

After changing the value and saving the file, restart the `ssh` service, as shown here:

```
$ sudo service ssh restart
```

Now, here is the output when I try to log in without the key:
Chapter 1

The firewall rules
For the servers and clients, the rule for security is: "Deny all incoming connections and allow only those needed." Here are the steps:

- Check which ports to keep open
- Define the firewall rules
- Make the firewall rules persistent

Checking which ports to keep open
We will use `netstat` to check the listening ports and running services. Here, we already know that for administrative purposes, we need to keep the SSH port 22 open. But we also need to check other ports that Puppet and Foreman are using. The command to check the listening ports and services is as follows:

```
$ sudo netstat -nlput
```

- `netstat`: The command to check network connections.
- `n` flag: This shows addresses in the numeric format
- `l` flag: This shows only the listening ports
- `p` flag: This shows the PID name of the program that the socket belongs to
- `u` flag: This shows the UDP ports
- `t` flag: This shows the TCP ports

```
puppet@puppetmaster:~$ sudo netstat -nlput
[sudo] password for puppet:
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local AddressForeign Address State PID/Program name
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 884/sshd
tcp 0 0 127.0.0.1:5432 0.0.0.0:* LISTEN 891/postgres
tcp 0 0 0.0.0.0:8443 0.0.0.0:* LISTEN 1824/ruby
tcp 0 0 127.0.0.1:40005 0.0.0.0:* LISTEN 1673/foreman
tcp 0 0 127.0.0.1:43946 0.0.0.0:* LISTEN 1531/rack
tcp6 0 0 :::80  :::* LISTEN 1401/apache2
tcp6 0 0 :::22  :::* LISTEN 884/sshd
tcp6 0 0 :::5432  :::* LISTEN 891/postgres
tcp6 0 0 :::443  :::* LISTEN 1401/apache2
tcp6 0 0 :::8140  :::* LISTEN 1401/apache2
udp6 0 0 :::69  :::* 832/in.tftpd
```
Installing Puppet Server and Foreman

Here are the ports that we need to keep open:

- **22**: ssh
- **80,445**: HTTP and HTTPS ports to connect Foreman
- **8443**: Foreman proxy is running on this port as a proxy
- **8140**: Puppet Master listens to this port

You can also remember that in the section *Installing Foreman* when the installation finishes, it gives the details about the **8443** and **8140** ports.

Defining firewall rules

We will use iptables as the firewall. It comes preinstalled on the Ubuntu Server. First, let’s check the rules that we have. If no rules are defined previously, all the policies will be in the **ACCEPT** state. The commands to check the iptables rules are as follows:

```bash
$ sudo iptables -L -v
```

- **iptables**: This is the command to manage the firewall
- **L** flag: This lists all rules
- **v** flag: Verbose output. This shows the rule options and packet counters

<table>
<thead>
<tr>
<th>puppet@puppetmaster:~$ sudo iptables -L -v</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain INPUT (policy ACCEPT 156 packets, 136K bytes)</td>
</tr>
<tr>
<td>pkts bytes target prot opt in out source</td>
</tr>
<tr>
<td>Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)</td>
</tr>
<tr>
<td>pkts bytes target prot opt in out source</td>
</tr>
<tr>
<td>Chain OUTPUT (policy ACCEPT 150 packets, 137K bytes)</td>
</tr>
<tr>
<td>pkts bytes target prot opt in out source</td>
</tr>
</tbody>
</table>

Allowing ingress traffic for the SSH port 22

We will accept traffic from any source when the destination port is port 22. Here is the command:

```bash
$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

- **-A** flag is used for adding rules.
- **-p tcp**, here **p** flag is for the protocol definition and **TCP** is the protocol.
• **--dport 22**, here, dport is the destination port definitions and the port is 22.
• **-j ACCEPT**, here, j flag tells what to do. Here, we accept the packet, if it matches the rule.

**Allowing ingress traffic for HTTP port 80**
We will accept traffic from any source when the destination port is port 80. Here is the command:

```
$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

**Allowing ingress traffic for HTTPS port 443**
We will accept traffic from any source when the destination port is port 443. Here is the command:

```
$ sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

**Allowing ingress traffic for Foreman proxy port 8443**
We will accept traffic from any source when the destination port is port 8443. Here is the command:

```
$ sudo iptables -A INPUT -p tcp --dport 8443 -j ACCEPT
```

**Allowing ingress traffic for Puppetmaster port 8140**
We will accept traffic from any source when the destination port is port 8140. Here is the command:

```
$ sudo iptables -A INPUT -p tcp --dport 8140 -j ACCEPT
```

**Allowing all that is established from us**
We need to define this rule. Otherwise, any traffic connection will not be complete. We will be able to send traffic outside, but never be able to get answers back. So, we will allow any incoming packet that is related to our outgoing traffic.

```
$ sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

• **-m**: This flag is used to match certain conditions. It can be used with different types of modules.
• **state --state ESTABLISHED,RELATED**: state is the module name that checks the statuses of a connection. Here, we allow any connection that is related to the established and related connections.
Denying all the incoming traffic

This time, we do not give any protocol or port number. So, this means that all the input traffic will be dropped.

$ sudo iptables -P INPUT DROP

After completing the rules, let’s check the iptables rules again:

<p>| Chain INPUT (policy ACCEPT 0 packets, 0 bytes)                                      |</p>
<table>
<thead>
<tr>
<th>pkeys bytes target</th>
<th>prot opt in</th>
<th>out source</th>
<th>destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>215 1392B ACCEPT</td>
<td>tcp --</td>
<td>any any anywhere</td>
<td>anywhere tcp dpt:ssh</td>
</tr>
<tr>
<td>0 ACCEPT</td>
<td>tcp --</td>
<td>any any anywhere</td>
<td>anywhere tcp dpt:http</td>
</tr>
<tr>
<td>0 ACCEPT</td>
<td>tcp --</td>
<td>any any anywhere</td>
<td>anywhere tcp dpt:https</td>
</tr>
<tr>
<td>0 ACCEPT</td>
<td>tcp --</td>
<td>any any anywhere</td>
<td>anywhere tcp dpt:8443</td>
</tr>
<tr>
<td>6 852 ACCEPT</td>
<td>all --</td>
<td>any any anywhere</td>
<td>anywhere state RELATED,ESTABLISHED</td>
</tr>
<tr>
<td>0 DROP</td>
<td>all --</td>
<td>any any anywhere</td>
<td>anywhere</td>
</tr>
</tbody>
</table>

| Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)                                     |
| pkeys bytes target       | prot opt in | out source | destination      |

As you can see in the preceding screenshot, all the rules are defined in the order that we defined.

IMPORTANT:

Define the “deny all” rule last. The order of the rules is important. The first rule will be applied first. So, if you first define the “deny all” rule, you will not be able to connect with SSH and your connection will drop.

Making the iptables rules persistent

The rules we defined are not persistent. So whenever you restart your server, the rules will be lost. To prevent this, we will install iptables-persistent. This software will keep our rules and will enable them at the startup. First install it using the following command:

$ sudo apt-get install iptables-persistent

While installing, it will ask you to save the current configuration to a file name such as /etc/iptables/rules.v4. Answer this with a yes. The second question will be about IPv6. We did not define any rules for it, so answer no for this.

After completing the setup, reboot your server and list the rules of iptables to see that they are still there.
Summary

In this chapter, we started by learning what Puppet is. After this, we continued with the differences between Puppet implementations. We got hands-on experience by installing Puppet Server and Foreman. The final step was learning about how to keep your server secure. In the next chapter, we will deal with the Puppet agents and their installation on the hosts.
Where to buy this book
You can buy Learning Puppet for Windows Server from the Packt Publishing website. Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers. Click here for ordering and shipping details.