**Steps for Setting up and Running sudo**

**Don’t touch empty file in sudoers.d folder, everything goes haywire**

1. You will need to make modifications as the administrative or super user, root. While **sudo** will become the preferred method of doing this, we don’t have it set up yet, so we will use **su** (which we will discuss later in detail) instead. At the command line prompt, type **su**and press **Enter.** You will then be prompted for the root password, so enter it and press **Enter**. You will notice that nothing is printed; this is so others cannot see the password on the screen. You should end up with a different looking prompt, often ending with ‘#’. For example: $ su Password: #
2. Now you need to create a configuration file to enable your user account to use **sudo**. Typically, this file is created in the /etc/sudoers.d/ directory with the name of the file the same as your username. For example, for this demo, let’s say your username is “student”. After doing step 1, you would then create the configuration file for “student” by doing this: # echo "student ALL=(ALL) ALL" > /etc/sudoers.d/student
3. Finally, some Linux distributions will complain if you don’t also change permissions on the file by doing: # chmod 440 /etc/sudoers.d/student

Or try an entry in /etc/sudoers

#visudo

Sudo –i

<https://www.digitalocean.com/community/tutorials/how-to-edit-the-sudoers-file-on-ubuntu-and-centos>

**ser Privilege Lines**

The fourth line, , which dictates the root user's sudo privileges, is different from the preceding lines. Let's take a look at what the different fields mean:

* root ALL=(ALL:ALL) ALL  
  The first field indicates the username that the rule will apply to (root).
* demo ALL=(ALL:ALL) ALL  
  The first "ALL" indicates that this rule applies to all hosts.
* demo ALL=(ALL:ALL) ALL  
  This "ALL" indicates that the root user can run commands as all users.
* demo ALL=(ALL:ALL) ALL  
  This "ALL" indicates that the root user can run commands as all groups.
* demo ALL=(ALL:ALL) ALL  
  The last "ALL" indicates these rules apply to all commands.

This means that our root user can run any command using sudo, as long as they provide their password.

**Group Privilege Lines**

The next two lines are similar to the user privilege lines, but they specify sudo rules for groups.

Names beginning with a "%" indicate group names.

Here, we see the "admin" group can execute any command as any user on any host. Similarly, the sudo group can has the same privileges, but can execute as any group as well.

**Included /etc/sudoers.d Line**

The last line might look like a comment at first glance:

/etc/sudoers

. . .

#includedir /etc/sudoers.d

It *does* begin with a #, which usually indicates a comment. However, this line actually indicates that files within the /etc/sudoers.d directory will be sourced and applied as well.

Files within that directory follow the same rules as the /etc/sudoers file itself. Any file that does not end in ~ and that does not have a . in it will be read and applied to the sudo configuration.

This is mainly meant for applications to alter sudo privileges upon installation. Putting all of the associated rules within a single file in the /etc/sudoers.d directory can make it easy to see which privileges are associated with which accounts and to reverse credentials easily without having to try to manipulate the /etc/sudoers file directly.

As with the /etc/sudoers file itself, you should always edit files within the /etc/sudoers.d directory with visudo. The syntax for editing these files would be:

* sudo visudo -f /etc/sudoers.d/file\_to\_edit

**How To Give a User Sudo Privileges**

The most common operation that users want to accomplish when managing sudo permissions is to grant a new user general sudo access. This is useful if you want to give an account full administrative access to the system.

The easiest way of doing this on a system set up with a general purpose administration group, like the Ubuntu system in this guide, is actually to just add the user in question to that group.

For example, on Ubuntu 16.04, the sudo group has full admin privileges. We can grant a user these same privileges by adding them to the group like this:

* sudo usermod -aG sudo username

The gpasswd command can also be used:

* sudo gpasswd -a username sudo

These will both accomplish the same thing.

On CentOS, this is usually the wheel group instead of the sudo group:

* sudo usermod -aG wheel username

Or, using gpasswd:

* sudo gpasswd -a username wheel

On CentOS, if adding the user to the group does not work immediately, you may have to edit the /etc/sudoers file to uncomment the group name:

* sudo visudo

/etc/sudoers

. . .

%wheel ALL=(ALL) ALL

. . .

**How To Set Up Custom Rules**

Now that we have gotten familiar with the general syntax of the file, let's create some new rules.

**How To Create Aliases**

The sudoers file can be organized more easily by grouping things with various kinds of "aliases".

For instance, we can create three different groups of users, with overlapping membership:

/etc/sudoers

. . .

User\_Alias GROUPONE = abby, brent, carl

User\_Alias GROUPTWO = brent, doris, eric,

User\_Alias GROUPTHREE = doris, felicia, grant

. . .

Group names must start with a capital letter. We can then allow members of GROUPTWO to update the apt database by creating a rule like this:

/etc/sudoers

. . .

GROUPTWO ALL = /usr/bin/apt-get update

. . .

If we do not specify a user/group to run as, as above, sudo defaults to the root user.

We can allow members of GROUPTHREE to shutdown and reboot the machine by creating a "command alias" and using that in a rule for GROUPTHREE:

/etc/sudoers

. . .

Cmnd\_Alias POWER = /sbin/shutdown, /sbin/halt, /sbin/reboot, /sbin/restart

GROUPTHREE ALL = POWER

. . .

We create a command alias called POWER that contains commands to power off and reboot the machine. We then allow the members of GROUPTHREE to execute these commands.

We can also create "Run as" aliases, which can replace the portion of the rule that specifies the user to execute the command as:

/etc/sudoers

. . .

Runas\_Alias WEB = www-data, apache

GROUPONE ALL = (WEB) ALL

. . .

This will allow anyone who is a member of GROUPONE to execute commands as the www-data user or the apache user.

Just keep in mind that later rules will override earlier rules when there is a conflict between the two.

**How To Lock Down Rules**

There are a number of ways that you can achieve more control over how sudo reacts to a call.

The updatedb command associated with the mlocate package is relatively harmless on a single-user system. If we want to allow users to execute it with root privileges *without* having to type a password, we can make a rule like this:

/etc/sudoers

. . .

GROUPONE ALL = NOPASSWD: /usr/bin/updatedb

. . .

NOPASSWD is a "tag" that means no password will be requested. It has a companion command called PASSWD, which is the default behavior. A tag is relevant for the rest of the rule unless overruled by its "twin" tag later down the line.

For instance, we can have a line like this:

/etc/sudoers

. . .

GROUPTWO ALL = NOPASSWD: /usr/bin/updatedb, PASSWD: /bin/kill

. . .

Another helpful tag is NOEXEC, which can be used to prevent some dangerous behavior in certain programs.

For example, some programs, like "less", can spawn other commands by typing this from within their interface:

!command\_to\_run

This basically executes any command the user gives it with the same permissions that "less" is running under, which can be quite dangerous.

To restrict this, we could use a line like this:

/etc/sudoers

. . .

username ALL = NOEXEC: /usr/bin/less

. . .

**Miscellaneous Information**

There are a few more pieces of information that may be useful when dealing with sudo.

If you specified a user or group to "run as" in the configuration file, you can execute commands as those users by using the "-u" and "-g" flags, respectively:

* sudo -u run\_as\_user command
* sudo -g run\_as\_group command

For convenience, by default, sudo will save your authentication details for a certain amount of time in one terminal. This means you won't have to type your password in again until that timer runs out.

For security purposes, if you wish to clear this timer when you are done running administrative commands, you can run:

* sudo -k

If, on the other hand, you want to "prime" the sudo command so that you won't be prompted later, or to renew your sudo lease, you can always type:

* sudo -v

You will be prompted for your password, which will be cached for later sudo uses until the sudo time frame expires.

If you are simply wondering what kind of privileges are defined for your username, you can type:

* sudo -l

This will list all of the rules in the /etc/sudoers file that apply to your user. This gives you a good idea of what you will or will not be allowed to do with sudo as any user.

There are many times when you will execute a command and it will fail because you forgot to preface it with sudo. To avoid having to re-type the command, you can take advantage of a bash functionality that means "repeat last command":

* sudo !!

The double exclamation point will repeat the last command. We preceded it with sudo to quickly change the unprivileged command to a privileged command.

For some fun, you can add the following line to your /etc/sudoers file with visudo:

* sudo visudo

/etc/sudoers

. . .

Defaults insults

. . .

This will cause sudo to return a silly insult when a user types in an incorrect password for sudo. We can use sudo -k to clear the previous sudo cached password to try it out:

* sudo -k
* sudo ls

Output

[sudo] password for demo: # enter an incorrect password here to see the results

Your mind just hasn't been the same since the electro-shock, has it?

[sudo] password for demo:

My mind is going. I can feel it.