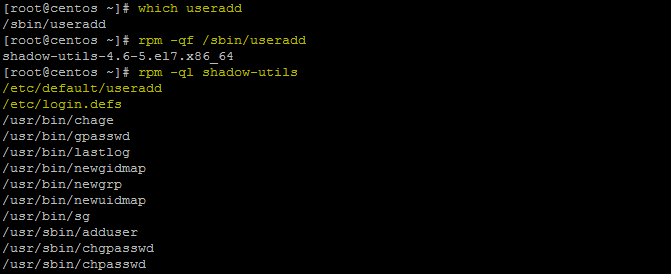
**Linux-User & Group Management**

Super user/Administartor/root: UID is **0 (zero)** (created from installation time)

Normal user: UID **1000 to 65000** (created by root)

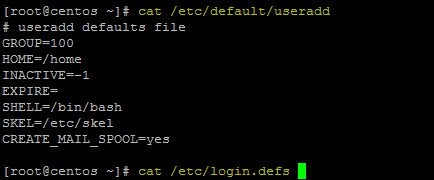
System user: UID **1 to 999** (created by service)

**# adduser wasim**

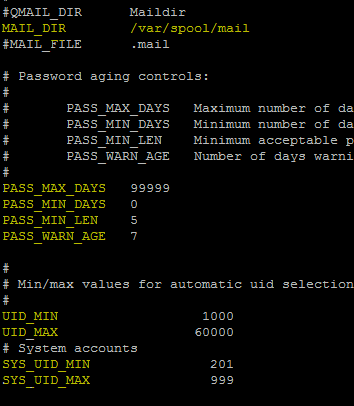
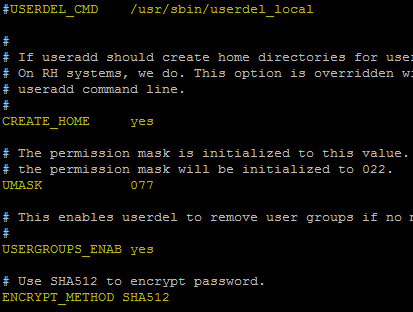


We can set or change for new user default setting from **/etc/default/useradd**

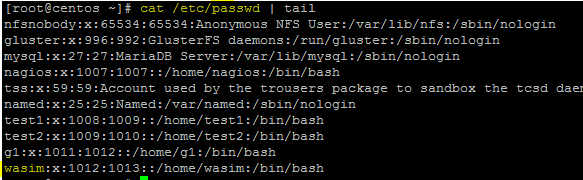
File for future created user.



We can set or change for new user default setting from **/etc/login.defs** like permission id password policy.

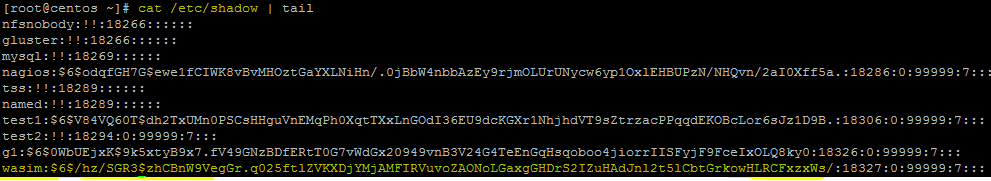
 

User add entry or list save in **/etc/passwd** file

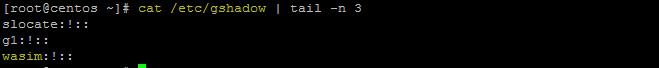


[username]:[x]:[UID]:[GID]:[Comment]:[Homedirectory]:[Default shell]

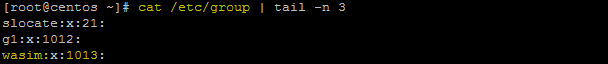
User password entry or list save in **/etc/shadow** file



User group entry or list save in **/etc/gshadow** file

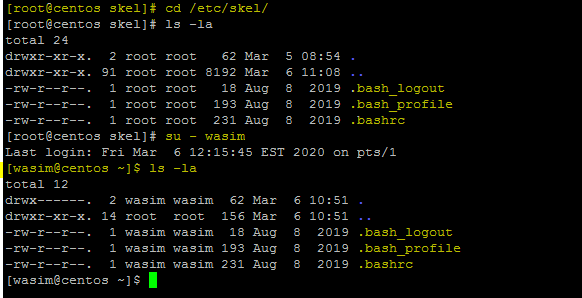


Default user group by user name or list save in **/etc/group** file

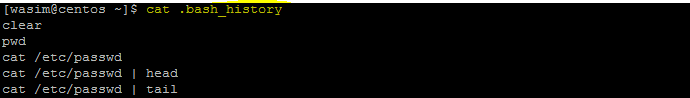


[Group name]:[Group password]:[GID]:[Group members]

User home directory copy from SKELL directory entry or list save in **/etc/skell** file.When user created some **hidden file** also create in each and every user home directory and it will copy from **/etc/skel/** directory**.**



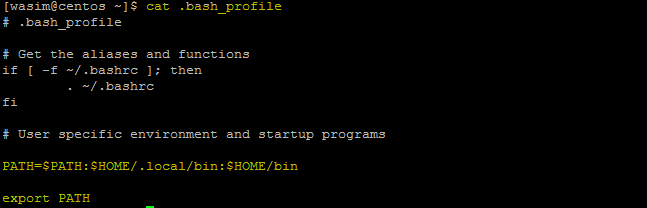
Command check run by user in **.bash\_history** file



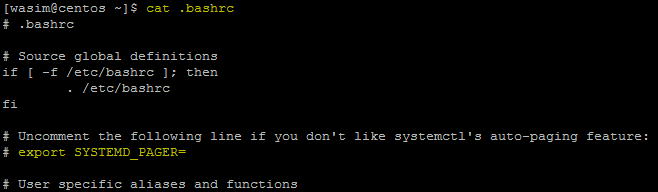
The **~/.bash\_logout** file contains instructions for the logout procedure. User Logout shell



When we login in LINUX machine the **.bash\_profile** well execute and for root **/etc/profile** will execute.

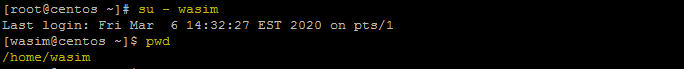


After login in LINUX machine while we open new terminal then **.bsahrc** file will execute.

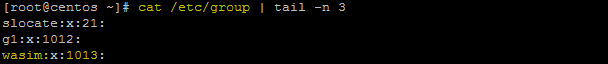


While creating any user by default it will create for user:

**Home Directory**



**Group**



**Login shell**



**Custom user set**

**# useradd -u 2000 username Assign UID**

**# useradd -g username Assign Primary group**

**# useradd -G g1,g2,g3 username Assign Secondary group**

**# useradd -c “Normal user” username Assign Comment**

**# useradd -s /bin/bash username Assign shell**

**# useradd -d /opt/rhch username Assign home directory**

**# useradd -e 2020-08-28 username Assign expire date**

**# useradd -p Welcome@123 username Assign manual password**

**Manage User**

**# usermod -l username username1 User name change**

**# usermod -aG g1,g2 username Add to group**

**# usermod -g groupname username Primary group change**

**# usermod -s /sbin/nologin username No-loginshell Unable to login user**

**# usermod -L username User Lock**

**# usermod -U username User Unlock**

**# usermod -f username User Inactive**

**# usermod -e 2020-08-27 username User Expire date**

**# usermod -o 1006 username Same UID set**

**User Account policy**

**# chage -l username Check user: Psswd/A.c/Exp-date/Passwd-warning**

**# chage -m 0 username Minimum number of password change set**

**# chage -M 90 username Maximum number of password change set**

**# chage -E 20-02-20 username Account expire date set**

**# chage -W 20 username Before Password expire date**

### Display SSH Warning Message to Users Before Login

To display **Welcome** or **Warning** message for **SSH** users before login. We use **issue.net** file to display a banner massages. Open the following file with **VI** editor.

# vi /etc/issue.net

Add the following banner sample message and save the file. You can add any custom banner message to this file.

######################################################### Welcome to Wasim.com All connections are monitored and recorded Disconnect IMMEDIATELY

Open the **master ssh** configuration file and enable banners.

# vi /etc/ssh/sshd\_config

Search for the word “**Banner**” and uncomment out the line and save the file.

#Banner /some/path

It should be like this.

Banner /etc/issue.net (you can use any path you want)

Next, restart the **SSH** daemon to reflect new changes.

# /etc/init.d/sshd restart

Stopping sshd: [ OK ]

Starting sshd: [ OK ]

### Display SSH Warning Message to Users After Login

To display banner messages after login, we use **motd** file, which is used to display banner massages after login. Now open it with **VI** editor.

# vi /etc/motd

Place the following banner sample message and save the file.

#########################################################Welcome to Wasim.com # All connections are monitored and recorded Disconnect IMMEDIATELY if you are not an authorized user! #########################################################

**SSH-Remote TMOUT (AutoLogOut)**

TMOUT variable in bash will terminate the shell if there is no activity for N seconds or you can set any time.

**Method 1:** Edit  **~/.bashrc**  or  **~/.bash\_profile**  file:

$ vi ~/.bashrc

$ vi ~/.bash\_profile

TMOUT=100

(Save and Exit)

$ export TMOUT=100 always it will be in seconds

$ echo $TMOUT To check who much time out set for user

$ unset TMOUT Remove time out set

**Method 2:** You can set auto logout for user

If we want to change permanently for future user we have to login as a root and create one file inside **/etc/profile.d/** directory.

**# vim /etc/profile.d/autologout.sh**

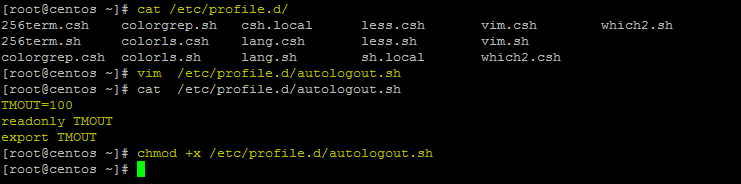
TMOUT=100 (Set time out seconds)

readonly TMOUT (Readonly no other user can’t change)

export TMOUT (export the command)

and give executable permission on file.

**# chmod +x /etc/profile.d/autologout.sh**



Please note that readonly variable can only be disabled by root in /etc/profile or ~/.bash\_profile.

**Method 3:** In this method, we will only making the SSH session users to log out after a particular period of inactivity.

Edit  **/etc/ssh/sshd\_config**  file:

$ sudo vi /etc/ssh/sshd\_config

Add/modify the following lines:

ClientAliveInterval 100

ClientAliveCountMax 0

(save and exit) Restart sshd service to take effect the changes.

$ sudo systemctl restart sshd

Now, ssh to this system from a remote system. After 100 seconds, the ssh session will be automatically closed and you will see the following message:

**LINUX Patching RPM and YUM**

**What Is Yum Server ?**

**The Yellowdog Updater, Modified (yum) is an open-source command-line package-management utility for Linux operating systems using the RPM Package Manager. Though yum has a command-line interface, several other tools provide graphical user interfaces to yum functionality. Yum allows automatic updates, package and dependency management, on RPM-based distributions. Like the Advanced Packaging Tool (APT) from Debian, yum works with software repositories (collections of packages), which can be accessed locally or over a network connection.**

**Advantages of Yum Server:**

**Automatic resolution of software dependencies. If a package installation or upgrade request is made and requires the installation or upgrade of additional packages, YUM can list these dependencies and prompt the user to install or upgrade them.**

**Command-line and graphical versions. The command-line version can be run on a system with a minimal number of software packages. The graphical versions offer ease-of-use and a user-friendly graphical interface to software management.**

**Multiple software locations at one time. YUM can be configured to look for software packages in more than one location at a time.**

**What is RPM?**

**RPM stands for Red Hat Package Manager which provides for installing, upgrading and removing packages on Linux distributions. It is a free software tool. Originally developed by Red Hat for Red Hat Linux, RPM is now used by many Linux distributions. It has also been ported to some other operating systems, such as Suse Linux and IBM’s AIX as of version**

**There are two main types of RPM packages: binary (or applications) and source.A binary RPM has been compiled for a particular architecture.Source code RPMs which provide the source code for other packages.The package name ending with suffix src.rpm such as:**

**Binary Package: xpdf-3.00-11.12.i386.rpm**

**Source code package: xpdf-3.00-11.12.src.rpm**

# rpm -ivh package\_name Install Rpm In Linux

# yum install package\_name

# rpm -ivh -nodeps package\_name Install RPM without Dependencies

# rpm -e package\_name Remove A Rpm

# yum remove package\_name

# rpm -ev -nodeps package\_name Remove RPM without Dependencies

# rpm -Uvh package\_name Upgrade The Existing Package Using Rpm Command

# yum update package\_ name

# rpm -qa package\_name | grep yum Check Installed Rpm

# yum search package\_name

# rpm -qpR package\_name Check Dependencies Of Rpm Package Before Installing

# rpm -checksig package\_name.rpm Check An Rpm Signature For A Package

# rpm -qa - -last | head List Recently Installed Rpm Packages

# yum list installed To List The installed package

# rpm -qf file\_location Query or find A File That Belongs Which Rpm Package

# yum provides file\_path

# rpm -qip package\_name Get The Information Of Rpm Package Before Installing

# yum info package\_name

# rpm -vp package\_name To verify the installed RPM

# rpm -qa gpg-pubkey package\_name To List All Imported Rpm Gpg Keys

# rpm -ql package\_name List All Files Of An Installed Rpm Package (query list)

# rpm -qc package\_name Display List Of Configuration Files For Specific Package

# rpm -qd package\_name Display List Of directory For Specific Package

# rpm -qL package\_name Display List of associated License file

# yum check-update Check The Updates For Yum Repository

# yum update To Update The Yum Repo

# yum repolist To List The Enabled Repositories

# yum repolist all

# yum check-update To Check the updates for yum repository

# yum list available To list all the available packages on a system

# yum list installed To list all the installed Packages on a system(@base)

# yum groupinstall 'haskell' To Install a Group of Package (say Haskell)

# yum grouplist List all available Group Packages

# yum groupupdate 'DNS Name Server' Update a Group Packages

# yum groupremove 'DNS Name Server' Remove a Group Packages

# yum --enablerepo=epel install phpmyadmin Install a Package from Specific Repository

# yum distro-sync We can sync all the packages on a system to stable release

# yum install deltarpm python-deltarpm createrepo To set up Local YUM repository

# yum history transaction id, login user who executed the particular action, date and time when the operation happened

# yum history list all

# yum history info httpd Delete and install time with transaction id full info

# yum history summary httpd transaction id full info

# yum history info transaction\_id(15) List full transaction id full info

# yum history package-info httpd epel-release Find Package Transaction Info

# yum history undo 2 will remove package W

# yum history redo 2 will reinstall package W

# yum history rollback 2 will remove packages from X, Y, and Z.

# yum history redo force-reinstall 16 reinstalls any packages that were installed in that transaction (via yum install, upgrade or downgrade)

# yum history redo force-remove 16 removes any packages that were updated or downgraded.

# yum -x python-3 update which you do not want to update use -x and --exclude

# yum --exclude httpd,php

# yum -y install yum-versionlock Disable Package Update Using versionlock Option

# yum versionlock add httpd Add package

# yum versionlock clear Remove from lock list

# yum versionlock list httpd Check lock list

# yum history addon-info will provide sources of additional information.

# yum history stats displays statistics about the current history DB.

# yum history sync enables us to alter the the rpmdb/yumdb data stored for any installed packages.

**Install Security Patches or Updates Automatically on CentOS and RHEL**

you will need to install the following package:

# yum update -y && yum install yum-cron -y

Once the installation is complete, open /etc/yum/yum-cron.conf and locate these lines – you will have to make sure that the values matches those listed here:

update\_cmd = change default to security

update\_messages = yes

download\_updates = yes

apply\_updates = yes

The first line indicates that the unattended update command will be:

# yum --security upgrade

whereas the other lines enable notifications and automatic download and installation of security upgrades.

The following lines are also required to indicate that notifications will be sent via email from root@localhost to the same account (again, you may choose another one if you want).

emit\_via = email

email\_from = root@localhost

email\_to = root

**Rebuilding corrupted RPM database on a CentOS/RHEL**

Let us verify integrity of the packages file. Make a backup of /var/lib/rpm/:  
# mkdir -vp /backups/rpm/  
# cp -avr /var/lib/rpm/\* /backups/rpm/  
Remove /var/lib/rpm/\_\_db\* files to avoid stale locks using rpm command:  
# cd /var/lib/rpm/  
# rm -vf \_\_db\*

Rebuild RPM database:  
# rpm --rebuilddb -vv  
# rpmdb\_verify Packages

**REPO-Configure**

**# mkdir testrepo**

**# cd /etc/yum.repo.d**

**# vi test.repo**

**[repo-name]**

**name=commend (Any thing you can)**

**baseurl=file:///testrepo (Directory path or if you have ftp then ftp path/link)**

**enable=1 (enable yum server 1 disable 0 )**

**gpgcheck=0 (package install time lenience key add and verify YES 1 or NO 0 )**

**# yum clean all**

**# yum repolist**

**# yum update**

**NFS-Network File System**

The NFS is a way of mounting Linux disk/directories over a network. An NFS server can export one or more directories that can be mounted on a remote machine data sharing.

**SERVER MACHINE**

1st Install NFS

# yum install nfs\* -y

2nd Create a nfs file transfer directory

# mkdir /testnfs

3rd Give the full permission to that file

# chmod 777 /testnfs

4th Go to Exports file and entry the that /testnfs

# vim /etc/exports

/testnfs \* (rw,sync)

(save&exit)

rw- Read and Write

sync- On the sport data will update

5th Restart the NFS service and enable for reboot time

# systemctl restart nfs-service

# systemctl enable nfs-service

6th NFS check in localserver

# exportfs -v

7th After any update or modification update export file

# exportfs -r

**CLIENT- MACHINE**

1st Before excessing any directory from client machine to server machine we have to create file in in localmachine.

# mkdir /clinet

2nd We have to check any NFS share directory available in network

# showmount -e nfs-server-IP

3rd We have to mount both share and created directory

# mount nfs-server-IP:/testnfs /client

4th Go to directory and create file

# cd /client

# mkdir f1{1..20}

**For Permanent Mount**

# vim /etc/fstab

Nfs-server-IP:/testnfs /client default nfs 0 0

**FTP-File Transfer Protocol**

**FTP is a commonly used protocol for transferring files between computer or it’s a centralize database of any organization**

It is support in LAN & WAN with same OS & different OS

Port number of FTP

**20** = Data connection for Uploading and Downloading

**21** = Connection control (client or ftp)

It allow two type of operation **GET** for Downloading & **PUL** for Uploading.

User: **Anonymous** (FTP installing time created) Home directories **/var/ftp** , Password not required for login, Allow only downloading.

User: **FTP** Home directories **/home** , Password required for login, Allow downloading and uploading.

**SERVER- MACHINE**

1st Install FTP package and restart & enable service vsftpd.

# yum install vsftpd\* -y

# systemctl restart vsftpd

# systemctl enable vsftpd

2nd Go to FTP configure file /var/ftp then go to pub file and create a directory for sharing and create some file inside that directory.

# cd /var/ftp

# cd /pub

# mkdir download

# cd download

# touch f1{1..10}

3rd Give full permission on ftp share path

# chmod 777 /var/ftp/pub

4th Active SE-Linux and restart vsftpd

# getsebool -a | grep ftp

Off (by default )

# setsebool -P ftp\_home\_directory on

# systemctl restart vsftpd

**CLIENT- MACHINE**

1st Install FTP package and check connection to server machine with Ping.

# yum install ftp\* -y

# ping server-machine-IP

2nd Check FTP working and login

# ftp ftp-server-machine-IP

Name : **Anonymous**

Password: Enter (No Password)

**ftp**> help

**ftp**> ls

**ftp**> cd pub

**ftp**> cd download

**ftp**> ls (all 10 file and download directory was showing)

**ftp**> get file-name (for download this file in client machine)

**ftp**> mget f1 f1

**ftp**> prompt (Interactive mode off for downloading all file )

**ftp**> mget f1\* (forcefully download all file)

**ftp**> lcd /tmp (downloading location change)

**ftp**> mget f1\* (now download in /tmp)

**ftp**> bye or exit

3rd By default FTP server all file downloading same location

# ftp ftp-server-IP

User: preet

Password: 1234

**ftp**> prompt (Interactive mode off for downloading all file )

**ftp**> mput f1\* (forcefully uploads all file)

**ftp**> exit

**SSH-Secure SHELL**

**SSH** (**Secure SHELL**) is an open source and most trusted network protocol that is used to login into remote servers for execution of commands and programs. It is also used to transfer files from one computer to another computer over the network using secure copy (**SCP**) Protocol.

##### **My Setup Environment**

SSH Client : 192.168.0.12 ( RHEL )

SSH Remote Host : 192.168.0.11 ( CentOS 7 )

If you are dealing with number of **Linux** remote servers, then **SSH Password-less** login is one of the best way to automate tasks such as automatic backups with scripts, synchronization files using scp and remote command execution.

In this example we will setup **SSH password-less** automatic login from server**192.168.0.12** as user **tecmint** to **192.168.0.11** with user **sheena**.

### Step 1: Create Authentication SSH-Kegen Keys on – (192.168.0.12)

First login into server **192.168.0.12** with user **tecmint** and generate a pair of public keys using following command.

**[tecmint@tecmint.com ~]$ ssh-keygen -t rsa**

Generating public/private rsa key pair.

Enter file in which to save the key (/home/tecmint/.ssh/id\_rsa): [Press enter key]

Created directory '/home/tecmint/.ssh'.

Enter passphrase (empty for no passphrase): [Press enter key]

Enter same passphrase again: [Press enter key]

Your identification has been saved in /home/tecmint/.ssh/id\_rsa.

Your public key has been saved in /home/tecmint/.ssh/id\_rsa.pub.

The key fingerprint is:

5f:ad:40:00:8a:d1:9b:99:b3:b0:f8:08:99:c3:ed:d3 tecmint@tecmint.com

The key's randomart image is:

+--[ RSA 2048]----+

| ..oooE.++|

| o. o.o |

| + + |

| +. |

+-----------------+

### Step 2: Create .ssh Directory on – 192.168.0.11

Use SSH from server **192.168.0.12** to connect server **192.168.0.11** using **sheena** as user and create **.ssh** directory under it, using following command.

**[tecmint@tecmint ~]$ ssh sheena@192.168.0.11 mkdir -p .ssh**

The authenticity of host '192.168.0.11 (192.168.0.11)' can't be established.

RSA key fingerprint is 45:0e:28:11:d6:81:62:16:04:3f:db:38:02:la:22:4e.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '192.168.0.11' (ECDSA) to the list of known hosts.

**sheena**@192.168.0.11's password: [**Enter Your Password Here**]

### Step 3: Upload Generated Public Keys to – 192.168.0.11

Use SSH from server **192.168.0.12** and upload new generated public key (**id\_rsa.pub**) on server **192.168.0.11** under **sheena**‘s **.ssh** directory as a file name **authorized\_keys**.

**[tecmint@tecmint ~]$ cat .ssh/id\_rsa.pub | ssh sheena@192.168.0.11 'cat >> .ssh/authorized\_keys'**

**sheena**@192.168.1.2's password: [**Enter Your Password Here**]

### Step 4: Set Permissions on – 192.168.0.11

Due to different SSH versions on servers, we need to set permissions on .ssh directory and authorized\_keys file.

**[tecmint@tecmint ~]$ ssh sheena@192.168.0.11 "chmod 700 .ssh; chmod 640 .ssh/authorized\_keys"**

**sheena**@192.168.0.11's password: [**Enter Your Password Here**]

### Step 5: Login from 192.168.0.12 to 192.168.0.11 Server without Password

From now onwards you can log into **192.168.0.11** as **sheena** user from server **192.168.0.12** as **tecmint** user without password.

**[tecmint@tecmint ~]$ ssh sheena@192.168.0.11**

### Copy SSH Key to 192.168.20.170 (tecmint)

Copying the key is a simple task and that can be completed by using **ssh-copy-id** command as shown.

# ssh-copy-id -i ~/.ssh/id\_rsa.pub user@remote-server

**Linux-File System**

## Install Nagios 4

We can manage n+ server in one tool health check.

### Install Build Dependencies

Because we are building Nagios Core from source, we must install a few development libraries that will allow us to complete the build.

First, install the required packages:

**sudo yum install gcc glibc glibc-common gd gd-devel make net-snmp openssl-devel xinetd unzip**

### Create Nagios User and Group

We must create a user and group that will run the Nagios process. Create a “nagios” user and “nagcmd” group, then add the user to the group with these commands:

**sudo useradd nagios**

**sudo groupadd nagcmd**

**sudo usermod -a -G nagcmd nagios**

Let’s install Nagios now.

### Install Nagios Core

Download the source code for the latest stable release of Nagios Core. Go to the [Nagios downloads page](http://www.nagios.org/download/core-stay-informed), and click the **Skip to download** link below the form. Copy the link address for the latest stable release so you can download it to your Nagios server.

At the time of this writing, the latest stable release is Nagios 4.1.1. Download it to your home directory with curl:

**cd ~**

**curl -L -O https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.1.1.tar.gz**

Extract the Nagios archive with this command:

**tar xvf nagios-\*.tar.gz**

Then change to the extracted directory:

**cd nagios-\***

Before building Nagios, we must configure it with this command:

**./configure --with-command-group=nagcmd**

Now compile Nagios with this command:

**make all**

Now we can run these make commands to install Nagios, init scripts, and sample configuration files:

**sudo make install**

**sudo make install-commandmode**

**sudo make install-init**

**sudo make install-config**

**sudo make install-webconf**

In order to issue external commands via the web interface to Nagios, we must add the web server user, apache, to the nagcmd group:

* **sudo usermod -G nagcmd apache**

### Install Nagios Plugins

Find the latest release of Nagios Plugins here: [Nagios Plugins Download](http://nagios-plugins.org/download/?C=M;O=D). Copy the link address for the latest version, and copy the link address so you can download it to your Nagios server.

At the time of this writing, the latest version is Nagios Plugins 2.1.1. Download it to your home directory with curl:

**cd ~**

**curl -L -O http://nagios-plugins.org/download/nagios-plugins-2.1.1.tar.gz**

Extract Nagios Plugins archive with this command:

**tar xvf nagios-plugins-\*.tar.gz**

Then change to the extracted directory:

**cd nagios-plugins-\***

Before building Nagios Plugins, we must configure it. Use this command:

**./configure --with-nagios-user=nagios --with-nagios-group=nagios --with-openssl**

Now compile Nagios Plugins with this command:

**Make**

Then install it with this command:

**sudo make install**

### Install NRPE

Find the source code for the latest stable release of NRPE at the [NRPE downloads page](http://sourceforge.net/projects/nagios/files/nrpe-2.x/). Download the latest version to your Nagios server.

At the time of this writing, the latest release is 2.15. Download it to your home directory with curl:

* **cd ~**
* **curl -L -O http://downloads.sourceforge.net/project/nagios/nrpe-2.x/nrpe-2.15/nrpe-2.15.tar.gz**

Extract the NRPE archive with this command:

* **tar xvf nrpe-\*.tar.gz**

Then change to the extracted directory:

* **cd nrpe-\***

Configure NRPE with these commands:

* **./configure --enable-command-args --with-nagios-user=nagios --with-nagios-group=nagios --with-ssl=/usr/bin/openssl --with-ssl-lib=/usr/lib/x86\_64-linux-gnu**

Now build and install NRPE and its xinetd startup script with these commands:

* **make all**
* **sudo make install**
* **sudo make install-xinetd**
* **sudo make install-daemon-config**

Open the xinetd startup script in an editor:

* **sudo vi /etc/xinetd.d/nrpe**

Modify the only\_from line by adding the private IP address of the your Nagios server to the end (substitute in the actual IP address of your server):

**only\_from = 127.0.0.1 191.168.1.101**

Save and exit. Only the Nagios server will be allowed to communicate with NRPE.

Restart the xinetd service to start NRPE:

* **sudo service xinetd restart**

Now that Nagios 4 is installed, we need to configure it.

## Configure Nagios

Now let’s perform the initial Nagios configuration. You only need to perform this section once, on your Nagios server.

### Organize Nagios Configuration

Open the main Nagios configuration file in your favorite text editor. We’ll use vi to edit the file:

**sudo vi /usr/local/nagios/etc/nagios.cfg**

Now find an uncomment this line by deleting the #:(Remove # sing)

**#cfg\_dir=/usr/local/nagios/etc/servers**

Save and exit.

Now create the directory that will store the configuration file for each server that you will monitor:

**sudo mkdir /usr/local/nagios/etc/servers**

### Configure Nagios Contacts

Open the Nagios contacts configuration in your favorite text editor. We’ll use vi to edit the file:

**sudo vi /usr/local/nagios/etc/objects/contacts.cfg**

Find the email directive, and replace its value (the highlighted part) with your own email address:

**email akramw574@gmail.com ; <<\*\*\*\*\* CHANGE THIS TO YOUR EMAIL ADDRESS \*\*\*\*\*\***

Save and exit.

### Configure check\_nrpe Command

Let’s add a new command to our Nagios configuration:

* **sudo vi /usr/local/nagios/etc/objects/commands.cfg**

Add the following to the end of the file:

**define command{**

**command\_name check\_nrpe**

**command\_line $USER1$/check\_nrpe -H $HOSTADDRESS$ -c $ARG1$**

**}**

Save and exit. This allows you to use the check\_nrpe command in your Nagios service definitions.

### Configure Apache

Use htpasswd to create an admin user, called “nagiosadmin”, that can access the Nagios web interface:

**sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin**

Enter a password at the prompt. Remember this login, as you will need it to access the Nagios web interface.

**Note:** If you create a user that is not named “nagiosadmin”, you will need to edit /usr/local/nagios/etc/cgi.cfg and change all the “nagiosadmin” references to the user you created.

Nagios is ready to be started. Let’s do that, and restart Apache:

**sudo systemctl daemon-reload**

**sudo systemctl start nagios.service**

**sudo systemctl restart httpd.service**

To enable Nagios to start on server boot, run this command:

**sudo chkconfig nagios on**

#### Optional: Restrict Access by IP Address

If you want to restrict the IP addresses that can access the Nagios web interface, you will want to edit the Apache configuration file:

**sudo vi /etc/httpd/conf.d/nagios.conf**

Find and comment the following two lines by adding # symbols in front of them:(Add # sing)

**Order allow,deny**

**Allow from all**

Then uncomment the following lines, by deleting the # symbols, and add the IP addresses or ranges (space delimited) that you want to allow to in the Allow from line:(Remove # sing)

**# Order deny,allow**

**# Deny from all**

**# Allow from 127.0.0.1**

As these lines will appear twice in the configuration file, so you will need to perform these steps once more.

Save and exit.

Now start Nagios and restart Apache to put the change into effect:

**sudo systemctl restart nagios.service**

**sudo systemctl restart httpd.service**

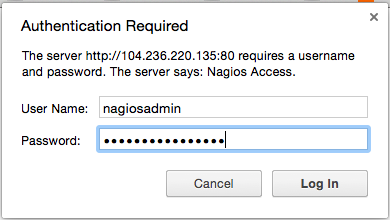
Nagios is now running, so let’s try and log in.

## Accessing the Nagios Web Interface

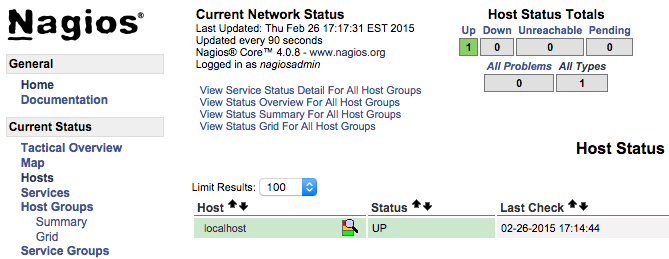
Open your favorite web browser, and go to your Nagios server (substitute the IP address or hostname for the highlighted part):

**http://nagios\_server\_public\_ip/nagios**

Because we configured Apache to use htpasswd, you must enter the login credentials that you created earlier. We used “**nagiosadmin**” as the username:



After authenticating, you will be see the default Nagios home page. Click on the **Hosts** link, in the left navigation bar, to see which hosts Nagios is monitoring:



As you can see, Nagios is monitoring only “localhost”, or itself.

Let’s monitor another host with Nagios!