Linux administrator should be familiar with **CLI** environment. Since **GUI** mode in Linux servers is not a common to be installed.
SSH
may the most popular protocol to enable Linux administrator to manage the servers via remote in secure way. Built-in with
SSH
command there is
SCP
command.
SCP
is used to copy file(s) between servers in secure way.

10 Linux SCP Commands

The below command will read as “**copy source_file_name**” into “**destination_folder**” at “**destination_host**” using “**username account**”.

Basic syntax of SCP

```
scp source_file_name username@destination_host:destination_folder
```

There are much parameters in **SCP** command that you can use. Here are the parameters that may useful on daily basis usage.

Provide the detail information of SCP process using -v parameter

Basic **SCP** command without parameter will copy the files in background. User will see nothing unless the process is done or some error appears. You can use “ **-v**” parameter to print debug information into the screen. It can help you debugging connection, authentication and configuration problems.

```
pungki@mint ~/Documents $ scp -v Label.pdf mrarianto@202.x.x.x:.
```

Sample Output

```
Executing: program /usr/bin/ssh host 202.x.x.x, user mrarianto, command scp -v -t . OpenSSH_6.0p1 Debian-3, OpenSSL 1.0.1c 10 May 2012 debug1: Reading configuration data /etc/ssh/ssh_config debug1: /etc/ssh/ssh_config line 19: Applying options for * debug1: Connecting to 202.x.x.x [202.x.x.x] port 22. debug1: Connection established. debug1: Host '202.x.x.x' is known and matches the RSA host key. debug1: Found key in /home/pungki/.ssh/known_hosts:1 debug1: ssh_rsa_verify: signature correct debug1: Next authentication method: password mrarianto@202.x.x.x's password: debug1: Authentication succeeded (password). Authenticated to 202.x.x.x ([202.x.x.x]:22). Sending file modes: C0770 3760348 Label.pdf Sink: C0770 3760348 Label.pdf Label.pdf 100% 3672KB 136.0KB/s 00:27 Transferred: sent 3766304, received 3000 bytes, in 65.2 seconds Bytes per second: sent 57766.4, received 46.0 debug1: Exit status 0
```

Provide modification times, access times, and modes from original files

The “**-p**” parameter will help you on this. An estimated time and the connection speed will appear on the screen.

```
pungki@mint ~/Documents $ scp -p Label.pdf mrarianto@202.x.x.x:.
```

Sample Output

```
mrarianto@202.x.x.x's password: Label.pdf 100% 3672KB 126.6KB/s 00:29
```

Make file transfer faster using -C parameter

One of parameter that can faster your file transfer is “**-C**” parameter. The “**-C**” parameter will compress your files on the go. The unique thing is the compression is only happen in the network. When the file is arrived to the destination server, it will returning into the original size as before the compression happen.

Take a look of these commands. It is using a single file of **93 Mb**.

```
  pungki@mint ~/Documents $ scp -pv messages.log mrarianto@202.x.x.x:.  Sample Output
  Executing: program /usr/bin/ssh host 202.x.x.x, user mrarianto, command scp -v -p -t .
OpenSSH_6.0p1 Debian-3, OpenSSL 1.0.1c 10 May 2012  debug1: Reading configuration data
/etc/ssh/ssh_config  debug1: /etc/ssh/ssh_config line 19: Applying options for *  debug1:
Connecting to 202.x.x.x [202.x.x.x] port 22.  debug1: Connection established.  debug1: identity
file /home/pungki/.ssh/id_rsa type -1  debug1: Found key in /home/pungki/.ssh/known_hosts:1
debug1: ssh_rsa_verify: signature correct  debug1: Trying private key: /home/pungki/.ssh/id_rsa
 debug1: Next authentication method: password  mrarianto@202.x.x.x's password:  debug1:
Authentication succeeded (password).  Authenticated to 202.x.x.x ([202.x.x.x]:22).  debug1:
Sending command: scp -v -p -t .  File mtime 1323853868 atime 1380425711  Sending file
timestamps: T1323853868 0 1380425711 0  messages.log 100% 93MB 58.6KB/s 27:05
Transferred: sent 97614832, received 25976 bytes, in 1661.3 seconds  Bytes per second: sent
58758.4, received 15.6  debug1: Exit status 0
```

Copying file without “**-C**” parameter will result **1661.3** second. Yo may compare the result to the command below which using “ **-C”** parameter.

```
  pungki@mint ~/Documents $ scp -Cpv messages.log mrarianto@202.x.x.x:.  Sample Output
  Executing: program /usr/bin/ssh host 202.x.x.x, user mrarianto, command scp -v -p -t .
OpenSSH_6.0p1 Debian-3, OpenSSL 1.0.1c 10 May 2012  debug1: Reading configuration data
/etc/ssh/ssh_config  debug1: /etc/ssh/ssh_config line 19: Applying options for *  debug1:
Connecting to 202.x.x.x [202.x.x.x] port 22.  debug1: Connection established.  debug1: identity
file /home/pungki/.ssh/id_rsa type -1  debug1: Host '202.x.x.x' is known and matches the RSA
host key.  debug1: Found key in /home/pungki/.ssh/known_hosts:1  debug1: ssh_rsa_verify:
signature correct  debug1: Next authentication method: publickey  debug1: Trying private key:
/home/pungki/.ssh/id_rsa  debug1: Next authentication method: password
mrarianto@202.x.x.x's password:  debug1: Enabling compression at level 6.  debug1:
Authentication succeeded (password).  Authenticated to 202.x.x.x ([202.x.x.x]:22).  debug1:
channel 0: new [client-session]  debug1: Sending command: scp -v -p -t .  File mtime
1323853868 atime 1380428748  Sending file timestamps: T1323853868 0 1380428748 0  Sink:
T1323853868 0 1380428748 0  Sending file modes: C0600 97517300 messages.log
messages.log 100% 93MB 602.7KB/s 02:38  Transferred: sent 8905840, received 15768 bytes,
in 162.5 seconds  Bytes per second: sent 54813.9, received 97.0  debug1: Exit status 0
debug1: compress outgoing: raw data 97571111, compressed 8806191, factor 0.09  debug1:
compress incoming: raw data 7885, compressed 3821, factor 0.48
```

As you can see, when you are using compression, transfer process is done in **162.5** second. It is **10** times faster than not using “**-C**” parameter. If you are copying a lot files across the network, “**-C**” parameter would help you to decrease the total time you need.

The thing that we should notice that compression method will not work on any files. When the source file is already compressed, you will not find any improvement there. Files such as **.zip**, **.rar**, **pictures**, and **.iso** files will not affected by “**-C**” parameter.

Select another cipher to encrypt files

By default **SCP** using “**AES-128**” to encrypt files. If you want to change to another cipher to encrypt it, you can use “**-c**” parameter. Take a look of this command.

```
  pungki@mint ~/Documents $ scp -c 3des Label.pdf mrarianto@202.x.x.x:.
mrarianto@202.x.x.x's password:  Label.pdf 100% 3672KB 282.5KB/s 00:13
```

Above command tell **SCP** to use **3des algorithm** to encrypt file. Please be careful that this parameter using “**-c**” not “**-C**“.

Limiting bandwidth usage

Another parameter that may useful is “**-l**” parameter. The “**-l**” parameter will limit the bandwidth to use. It will be useful if you do an automation script to copy a lot of file, but you don’t want the bandwidth is drained by the **SCP** process.

```
  pungki@mint ~/Documents $ scp -l 400 Label.pdf mrarianto@202.x.x.x:.
mrarianto@202.x.x.x's password:  Label.pdf 100% 3672KB 50.3KB/s 01:13
```

The **400** value behind “**-l**” parameter is mean that we limit the bandwidth for **SCP** process only **50 KB/sec**

. One thing to remember that bandwidth is specified in **Kilobits** / **sec** (**kbps**). It is mean that **8 bits** equal with **1 byte** .

While **SCP** counts in **Kilobyte**/**sec** (**KB/s**). So if you want to limit your bandwidth for **SCP** maximum only **50** **KB/s** , you need to set it into **50 x 8** = **400** .

Specify specific port to use with SCP

Usually **SCP** is using port **22** as a default port. But for security reason, you may change the port into another port. For example, we are using port **2249**. Then the command should be like this.

```
  pungki@mint ~/Documents $ scp -P 2249 Label.pdf mrarianto@202.x.x.x:.
mrarianto@202.x.x.x's password:  Label.pdf 100% 3672KB 262.3KB/s 00:14
```

Make sure that it use capital “**P**” not “**p**“, since “**p**” is already used for preserved times and modes.

Copy files inside directory recursively

Sometimes we need to copy directory and all **files** / **directories** inside it. It will be better if we can do it in **1** command. **SCP** support that scenario using “ **-r** ” parameter.

```
  pungki@mint ~/Documents $ scp -r documents mrarianto@202.x.x.x:.
mrarianto@202.x.x.x's password:  Label.pdf 100% 3672KB 282.5KB/s 00:13  scp.txt 100%
10KB 9.8KB/s 00:00
```

When the copy process is done, at the destination server you will found a directory named “**documents**” with all it’s files. The folder “**documents**” is automatically created.

Disable progress meter and warning / diagnostic message

If you choose not to see progress meter and warning / diagnostic messages from SCP, you may disable it using “**-q**” parameter. Here’s the example.

```
  pungki@mint ~/Documents $ scp -q Label.pdf mrarianto@202.x.x.x:.    mrarianto@202.x.x.x's
password:  pungki@mint ~/Documents $
```

As you can see, after the you enter the password, there is no any information about SCP process. After the process is complete, you will be see a prompt again.

Copy files using SCP through Proxy

Proxy server is usually used in office environment. Natively, SCP is not proxy configured. When your environment using proxy, you have to “tell” SCP to communicate with the proxy.

Here’s the scenario. The proxy address is **10.0.96.6** and the proxy port is **8080**. The proxy also implemented user authentication. First, you need to create “**~/.ssh/config”** file. Second you put this command inside it.

```
  ProxyCommand /usr/bin/corkscrew 10.0.96.6 8080 %h %p ~/.ssh/proxyauth
```

Then you need to create file “**~/.ssh/proxyauth**” which contain.

```
  myusername:mypassword
```

After that you can do SCP transparently as usual.

Please notice that corkscrew is might not installed yet on your system. On my Linux Mint, I need to install it first, using standard Linux Mint installation procedure.

```
$ apt-get install corkscrew
```

For other yum based systems, users can install corkscrew using the following yum command.

```
# yum install corkscrew
```

Another thing that since “~/**.ssh**/**proxyauth**” file contain your “**username**” and “**password**” in clear-text format, please make sure that the file can be accessed by you only.

Select different ssh_config file

For mobile user who often switch between company network and public network, it will be suffer to always change settings in SCP. It is better if we can put a different **ssh_config** file to match our needs.

Here’s a sample scenario

Proxy is used in company network but not in public network and you are regularly switch network.

```
pungki@mint ~/Documents $ scp -F /home/pungki/proxy_ssh_config Label.pdf
mrarianto@202.x.x.x:.  mrarianto@202.x.x.x's password:  Label.pdf 100% 3672KB 282.5KB/s
00:13
```

By default “**ssh_config**” file per user will be placed in “~/**.ssh**/**config**“. Creating a specific “**ssh _config**” file with proxy compatible, will make you easier to switch between networks.

When you are on company network, you can use “**-F**” parameter. When you are on public network, you can skip “**-F**” parameter.

That’s all about **SCP**. You can see **man pages** of **SCP** for more detail. Please feel free to leave comments and suggestions.