

# How To Run / Execute Command Using SSH

Vivek Gite



How do I run a command using ssh under UNIX, OS X, \*BSD, and Linux operating systems?

The SSH client program can be used for logging into a remote machine or server and for executing commands on a remote machine. When command is specified, it is executed on the remote host/server instead of a login shell. Let us see how to run and execute command using the ssh command on Linux, macOS, \*BSD or Unix-like systems.

Tutorial details	
Difficulty level	<a href="#">Easy</a>
Root privileges	No
Requirements	Linux or Unix terminal
Category	<a href="#">Terminal/ssh</a>
Prerequisites	ssh command
OS compatibility	BSD • <a href="#">Linux</a> • <a href="#">macOS</a> • <a href="#">Unix</a> • WSL
Est. reading time	4 minutes

Advertisement

## Run / Execute Command SSH Command Syntax

The syntax is as follows for executing commands over ssh:

```
$ ssh user1@server1 command1
$ ssh user1@server1 'command2'
# pipe #
$ ssh user1@server1 'command1 | command2'
# multiple commands (must enclose in quotes #
$ ssh admin@box1 "command1; command2; command3"
```

The ssh client will login to a server called server1, using user name called user1 and run a command call command1.

## Examples: Running commands over ssh

Get remote server date and time:

Find out remote server disk space usage:

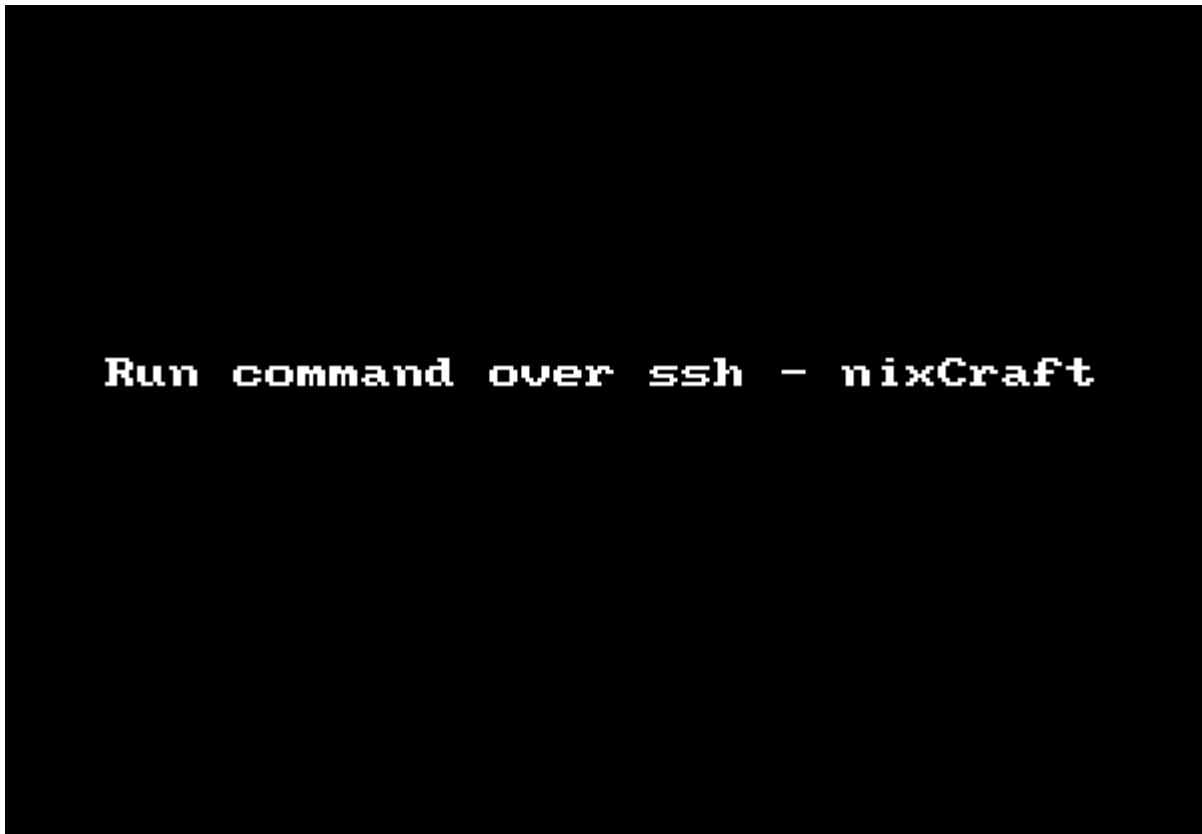
```
ssh user1@server1 'df -H'
```

Find out remote server kernel version and Linux distro names:

```
ssh root@nas01 "uname -mrs"
```

OR

```
ssh root@nas01 lsb_release -a
```



Animated gif 01: Running commands using ssh client

Run a script called /scripts/backup.sh:

```
ssh operator@oracle1 '/scripts/backup.sh'
```

Run sudo or su command using the following syntax:

```
## sudo syntax ##
```

```
ssh -t user@hostname sudo command
```

```
ssh -t user@hostname 'sudo command1 arg1 arg2'
```

```
## su syntax ##
```

```
ssh user@nas01 su -c "/path/to/command1 arg1 arg2"
```

```
# RHEL/CentOS specific #
```

```
ssh user@nas01 su --session-command="/path/to/command1 arg1 arg2"
```

```
ssh vivek@nixcraft.home.server su --session-command="/sbin/service httpd restart"
```

Without the -t option you will get an error that read as “[sudo: Sorry, you must have a tty to run sudo on a Linux and Unix](#)”.

## Running and executing multiple ssh command

Create a new file named commands.txt using the [cat command](#):

```
$ cat > commands.txt
```

Append command you wish to run:

Next execute commands remotely using ssh command from local file called commands.txt:

```
$ ssh server_name < commands.txt
```

```
$ ssh user@server_name < commands.txt
```

```
$ ssh admin@ls.backup < commands.txt
```

The screenshot shows a terminal window titled 'Terminal' with the following content:

```
[vivek@nixcraft-wks01 tmp]$ cat > commands.txt
date
uptime
df -H
[vivek@nixcraft-wks01 tmp]$ ssh admin@ls.backup < commands.txt
Pseudo-terminal will not be allocated because stdin is not a terminal.
Tue 23 Feb 2021 07:42:10 AM UTC
 07:42:10 up 16 days, 15:31,  0 users,  load average: 0.00, 0.00, 0.00
Filesystem      Size  Used Avail Use% Mounted on
udev            502M   0    502M   0% /dev
tmpfs           104M  11M   93M   11% /run
/dev/xvda1      43G   2.9G  38G    8% /
tmpfs           517M   0    517M   0% /dev/shm
tmpfs           5.3M   0    5.3M   0% /run/lock
tmpfs           517M   0    517M   0% /sys/fs/cgroup
/dev/xvdf       430G  339G   91G   79% /backup
tmpfs           104M   0   104M   0% /run/user/1000
[vivek@nixcraft-wks01 tmp]$
```

© www.cyberciti.biz

## How to run multiple ssh command when using shell scripts

The multi-line command syntax is as follows and need to take help of [Here document](#) feature provided by bash:

```
#!/bin/bash
```

```
_remote="ls.backup"
```

```
_user="vivek"
```

```
echo "Local system name: $HOSTNAME"
```

```
echo "Local date and time: $(date)"
```

```
echo
```

```
echo "**** Running commands on remote host named $_remote ****"
```

```
echo
```

```
ssh -T $_remote <<'EOL'
```

```
now="$(date)"
```

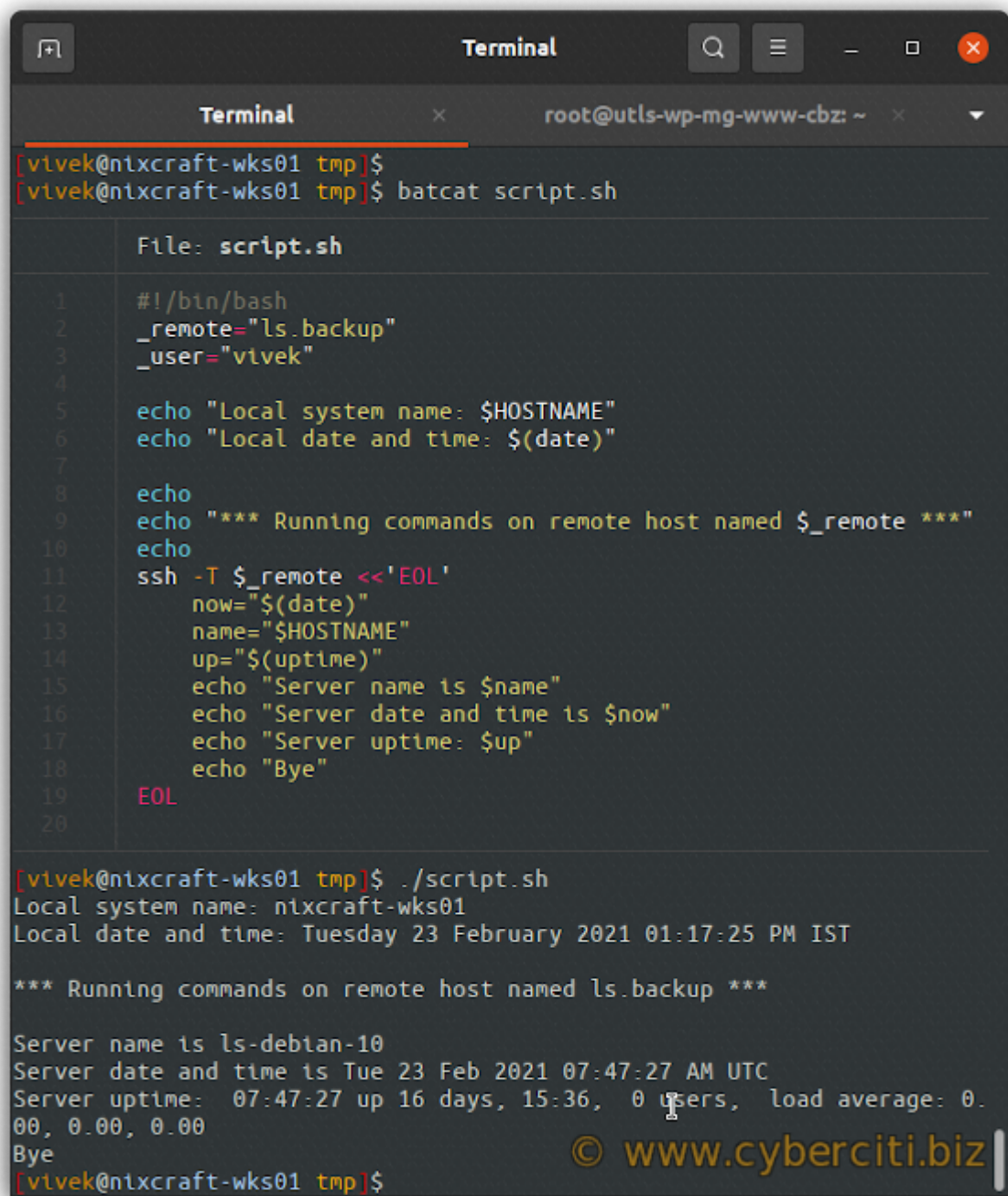
```

name="$HOSTNAME"
up="$(uptime)"
echo "Server name is $name"
echo "Server date and time is $now"
echo "Server uptime: $up"
echo "Bye"

```

EOL

Run the script:



The image shows a terminal window titled 'Terminal' with a tab for 'root@utls-wp-mg-www-cbz: ~'. The user 'vivek@nixcraft-wks01 tmp' runs the command 'batcat script.sh'. The terminal displays the contents of 'script.sh' with line numbers 1 through 20. The script sets environment variables, echoes local system information, and then uses 'ssh -T' to execute the same script on a remote host 'ls.backup'. The output shows the local system name as 'nixcraft-wks01' and the date and time as 'Tuesday 23 February 2021 01:17:25 PM IST'. The remote host output shows 'Server name is ls-debian-10', 'Server date and time is Tue 23 Feb 2021 07:47:27 AM UTC', and 'Server uptime: 07:47:27 up 16 days, 15:36, 0 users, load average: 0.00, 0.00, 0.00'. The terminal ends with 'Bye' and a copyright notice for 'www.cyberciti.biz'.

```

Terminal
root@utls-wp-mg-www-cbz: ~
[vivek@nixcraft-wks01 tmp]$
[vivek@nixcraft-wks01 tmp]$ batcat script.sh
File: script.sh
1  #!/bin/bash
2  _remote="ls.backup"
3  _user="vivek"
4
5  echo "Local system name: $HOSTNAME"
6  echo "Local date and time: $(date)"
7
8  echo
9  echo "*** Running commands on remote host named $_remote ***"
10 echo
11 ssh -T $_remote <<'EOL'
12     now="$(date)"
13     name="$HOSTNAME"
14     up="$(uptime)"
15     echo "Server name is $name"
16     echo "Server date and time is $now"
17     echo "Server uptime: $up"
18     echo "Bye"
19 EOL
20
[vivek@nixcraft-wks01 tmp]$ ./script.sh
Local system name: nixcraft-wks01
Local date and time: Tuesday 23 February 2021 01:17:25 PM IST

*** Running commands on remote host named ls.backup ***

Server name is ls-debian-10
Server date and time is Tue 23 Feb 2021 07:47:27 AM UTC
Server uptime: 07:47:27 up 16 days, 15:36, 0 users, load average: 0.
00, 0.00, 0.00
Bye
© www.cyberciti.biz
[vivek@nixcraft-wks01 tmp]$

```

Multi-line command executing using Heredoc feature of bash

## Summing up

## Various ways to execute commands remotely using SSH

Purpose	Syntax	Examples
Single-line ssh command	<code>ssh \$user@\$host command</code>	<code>ssh admin@ec2-server uptime</code>
Executing several commands	<code>ssh user@server "command1; command2; script1"</code>	<code>ssh vivek@linode-server "ls /etc/resolv.conf; date"</code>
Running a command with sudo	<code>ssh -t user@host sudo command</code>	<code>ssh -t jobs@backup sudo /scripts/backup.sh</code>
Run commands from a local file	<code>ssh user@hostname &lt; file</code>	<code>cat cmd.txt date git clone url cd project makessh vivek@ls.www-db-1 &lt; cmds.txt</code>
Use Heredoc bash feature to run many commands	<code>ssh -T \$HOST &lt;&lt; EOL</code>	<code>ssh -T admin@freebsdns &lt;&lt; EOL uptime date df -H echo "\$var" echo "\$HOSTNAME" EOL</code>
Multi-line command using Heredoc when you need to assign variables	<code>ssh -T box1&lt;&lt;'EOL'</code>	<code>ssh -T vivek@server1&lt;&lt;'ENDSSH' var=\$(date) echo "\$var" ENDSSH</code>

Please note that when you pass the `-T` to ssh when you wish to disable pseudo-terminal allocation. On the other hand, we can force pseudo-terminal allocation bypassing the `-t` option to ssh to execute arbitrary screen-based programs on a remote machine, which can be very useful. For instance, when implementing menu services. Multiple `-t` options force tty allocation, even if ssh has no local tty. See ssh [documentation](#) online or read it locally (offline) by typing the [man command](#) as follows:

```
$ man ssh
```