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	Easy
Root privileges	No
Requirements	Linux or Unix terminal
Category	Terminal/ssh
Prerequisites	ssh command
OS compatibility	BSD • Linux • macOS • Unix • WSL
Est. reading time	4 minutes

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:

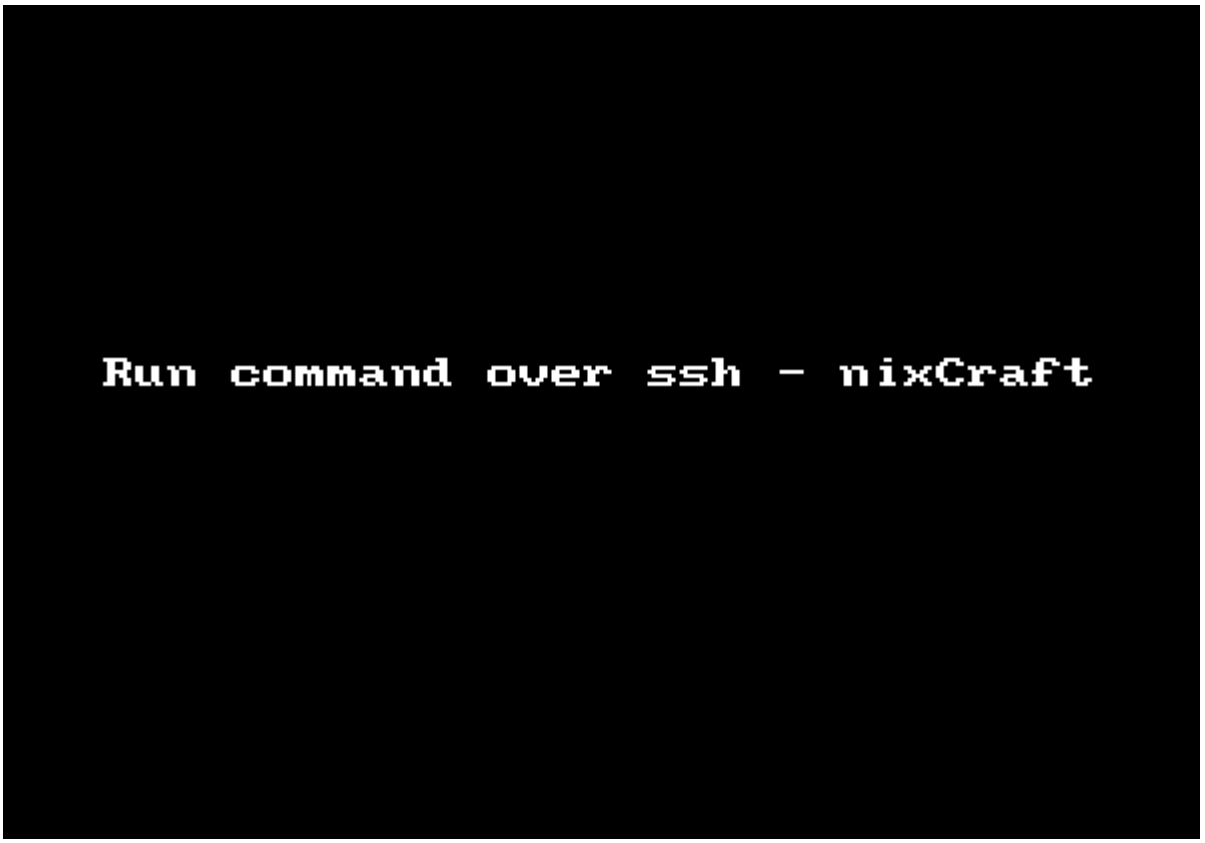
ssh user1@server1 'df -H'

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:

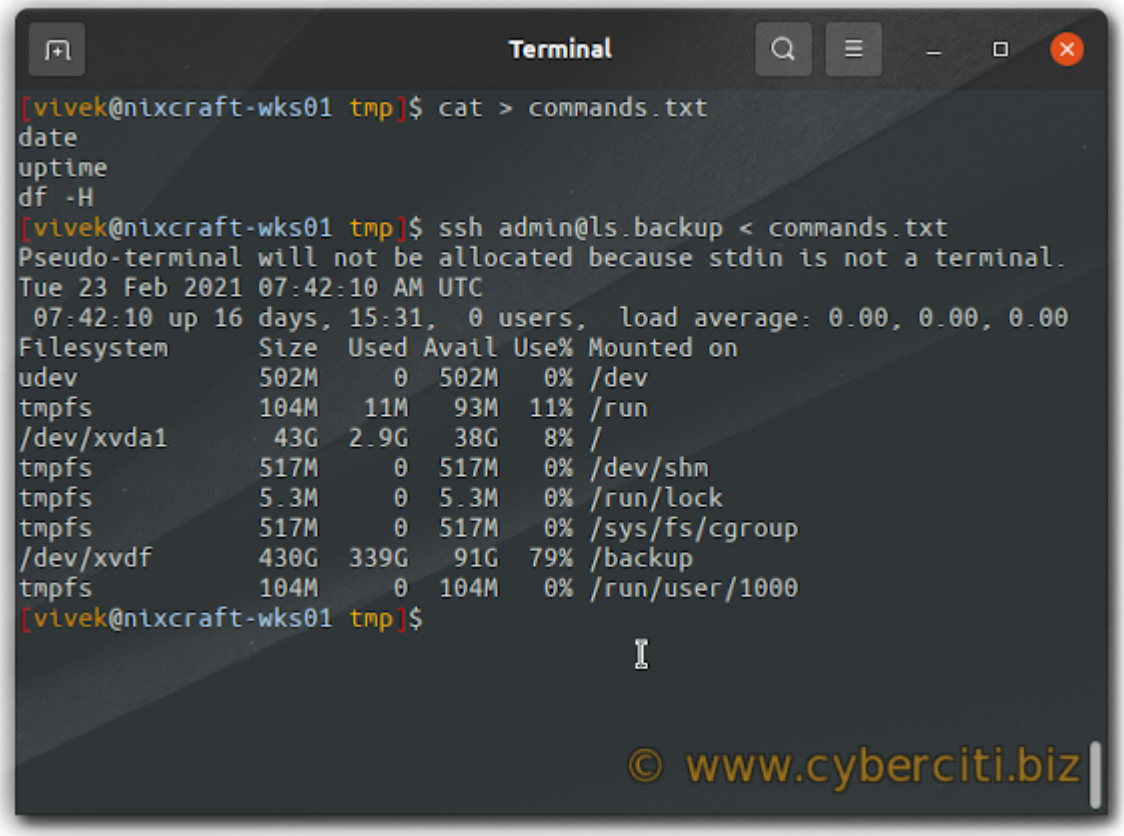
```
## sudo syntax ##
ssh -t user@hostname sudo command
ssh -t user@hostname 'sudo command1 arg1 arg2'


## 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
```



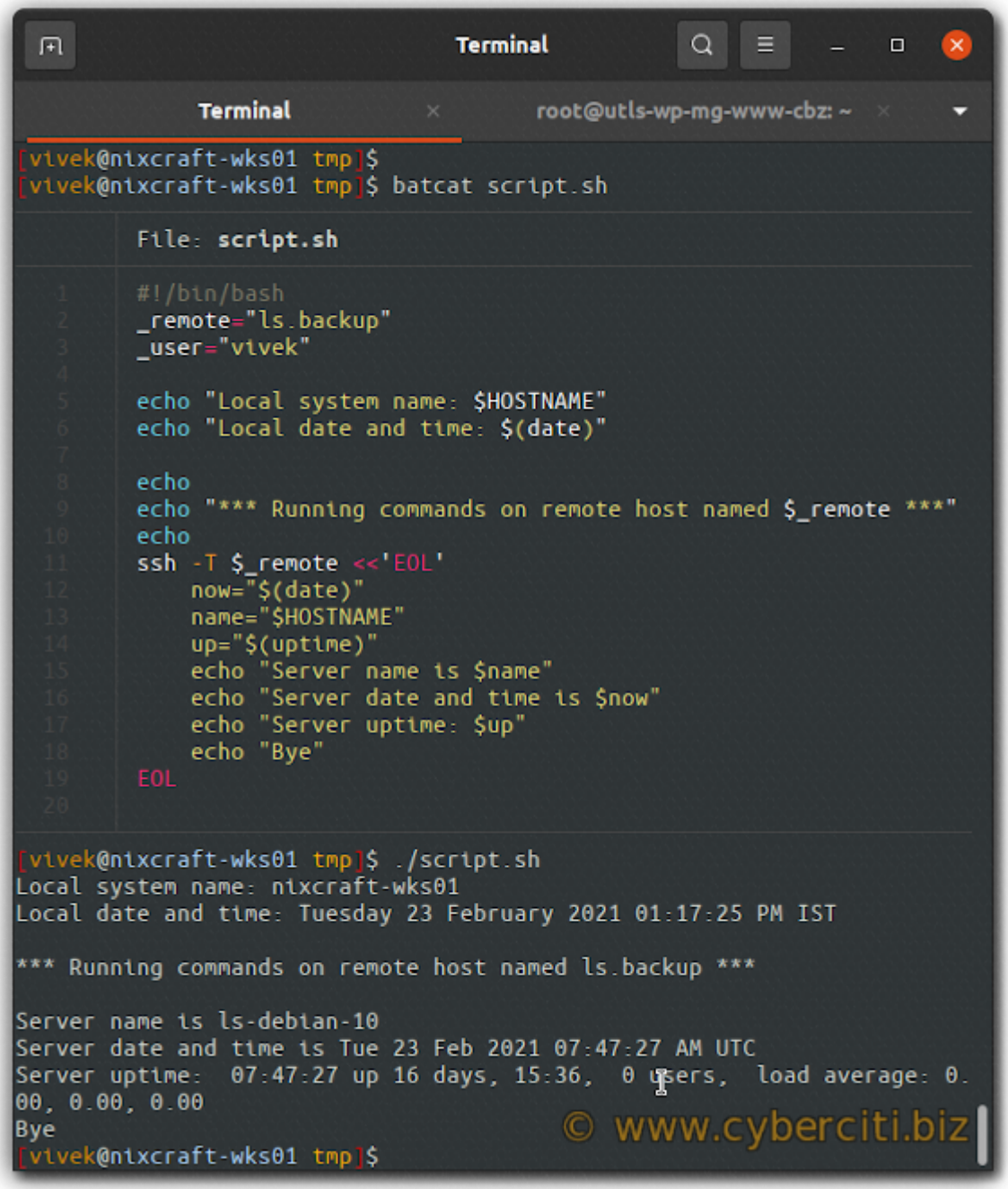
How to run multiple ssh command when using shell scripts

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:



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	`ssh $user@$host command`	`ssh admin@ec2-server uptime`
Executing several commands	`ssh user@server "command1; command2; script1"`	`ssh vivek@linode-server "ls /etc/resolv.conf; date"`
Running a command with sudo	`ssh -t user@host sudo command`	`ssh -t jobs@backup sudo /scripts/backup.sh`

Run commands from a local file	`ssh user@hostname < file`	`cat cmd.txt`
		`date`
		`git clone url`
		`cd project`
		`makessh vivek@ls.www-db-1 < cmds.txt`

Use Heredoc bash feature to run many commands	`ssh -T $HOST << EOL`	`ssh -T admin@freebsdnas <<`
		`EOL`
		`uptime`
		`date`
		`df -H`
		`echo "$var"`
		`echo "$HOSTNAME"`
		`EOL`

Multi-line command using Heredoc when you need to assign variables	`ssh -T box1<<'EOL'`	`ssh -T`
		`vivek@server1<<'ENDSSH'`
		`var=$(date)`
		`echo "$var"`
		`ENDSSH`

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
```