Monitor Network Connectivity using Bash Script

Prashant Lakhera Follow
Apr 12, 2018 · 3 min read

This is again one of the tasks which we used to deal it on a daily basis

The First step is to check what is by default available to us and the first utility that comes to my mind is ping

```
$ ping www.google.com

PING www.google.com (209.85.201.103): 56 data bytes

64 bytes from 209.85.201.103: icmp_seq=0 ttl=46 time=83.026 ms

64 bytes from 209.85.201.103: icmp_seq=1 ttl=46 time=83.929 ms

64 bytes from 209.85.201.103: icmp_seq=2 ttl=46 time=182.060 ms

^C
```

The Default behaviour of ping in Linux is to send icmp echo packet

indefinitely, while going through the ping help

```
$ ping --help

ping: unrecognized option `--help'

usage: ping [-AaDdfnoQqRrv] [-c count]
```

Ping accept an argument (-c count)

```
-c count

Stop after sending (and receiving) count ECHO_RESPONSE packets.
If this

option is not specified, ping will operate until interrupted.
If this

option is specified in conjunction with ping sweeps, each sweep
will

consist of count packets.
```

So let's restrict ping count to 3, if we got the response back then it mean our network connectivity is good

```
$ ping -c 3 www.google.com
```

```
PING www.google.com (172.217.7.132): 56 data bytes

64 bytes from 172.217.7.132: icmp_seq=0 ttl=55 time=80.352 ms

64 bytes from 172.217.7.132: icmp_seq=1 ttl=55 time=78.291 ms

64 bytes from 172.217.7.132: icmp_seq=2 ttl=55 time=78.926 ms

--- www.google.com ping statistics ---

3 packets transmitted, 3 packets received, 0.0% packet loss

round-trip min/avg/max/stddev = 78.291/79.190/80.352/0.862 ms
```

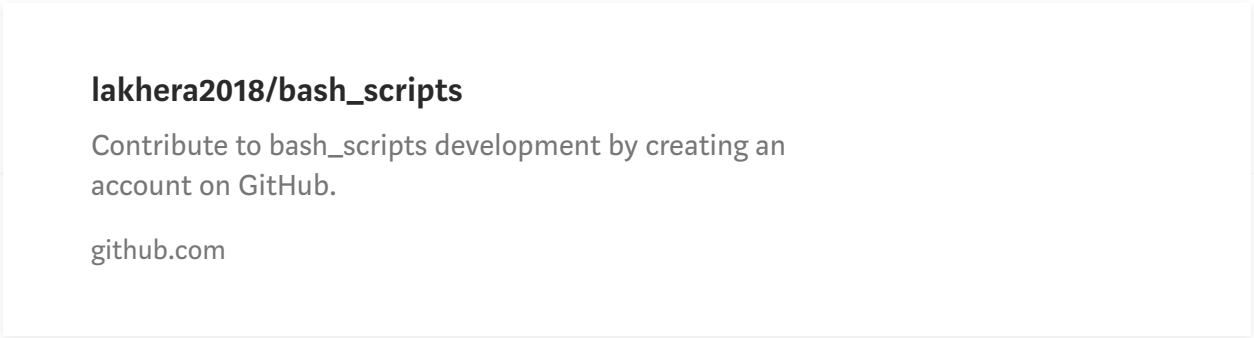
Now the next question is do we need this output as in case of shell scripts(then send it to /dev/null which act as a backhole) we are relying on exit code

```
$ ping -c 3 www.google.com > /dev/null
$ echo $?

0
```

Let's put together everything into basic shell script

```
#!/bin/bash

ping -c 3 www.google.com > /dev/null
```

```
if [ $? != 0 ]

then

echo "Your site seems to be down"

fi
```



Let's try with the host/website which we know for sure doesn't exist

```
#!/bin/bash

ping -c 3 www.donotexist.com > /dev/null

if [ $? != 0 ]

then

echo "Your site seems to be down"

fi
```

Gave your script execute permission

```
$ chmod +x networkcheck.sh
```

and test it

```
$ ./networkcheck.sh
Your site seems to be down
```

Now how can we improve this script, by adding a variable to our script

```
#!/bin/bash
MYSITE=www.google.com
ping -c 3 $MYSITE > /dev/null
if [ $? != 0 ]
then
echo `date +%F`
echo "Your site seems to be down"
fi
```

lakhera2018/bash_scripts

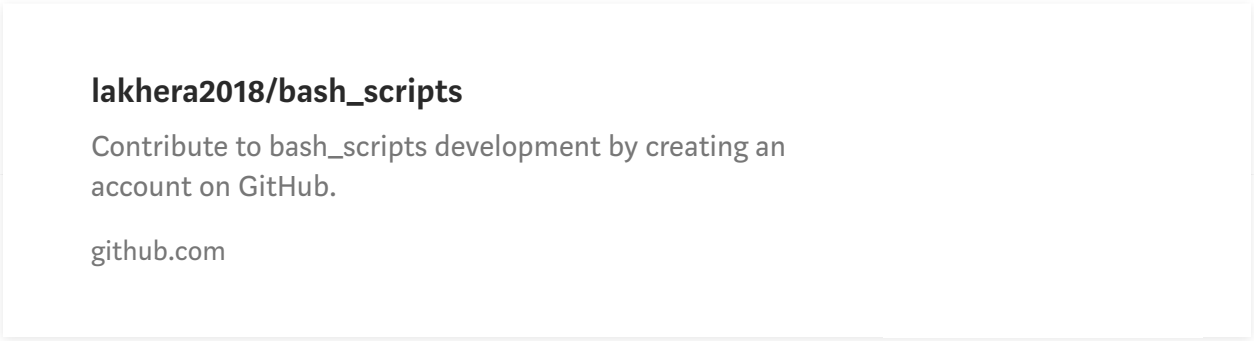
Contribute to bash_scripts development by creating an account on GitHub.

github.com

What else we can do to improve the script, we can add positional parameters

```
#!/bin/bash

if [ $# -ne 1 ]

then

echo "Please enter site name"

else

MYSITE=$1

fi

ping -c 3 $MYSITE > /dev/null

if [ $? != 0 ]

then

echo `date +%F`
```

```
echo "Your site seems to be down"

fi
```



In case of network scripts most of the time we used to give default values and if the user doesn't supply any value script will accept that default value, so this requires little modification in our existing script

```
#!/bin/bash

if [ $# -ne 1 ]

then

MYSITE=www.google.com

else

MYSITE=$1

fi

ping -c 3 $MYSITE > /dev/null
```

```
if [ $? != 0 ]

then

echo `date +%F`

echo "Your site seems to be down"

fi
```

Similarly we can send an email to individual/team in case of any issues(Usuall these type of scripts run via cron on daily basis)

```
#!/bin/bash

if [ $# -ne 1 ]

then

MYSITE=www.google.com

else

MYSITE=$1

fi

ping -c 3 $MYSITE > /dev/null

if [ $? != 0 ]

then
```

```
echo `date +%F`

echo "Your site seems to be down"

mail -s "Your $MYSITE seems to be down" user@gmail.com

fi
```

P.S: There some host/website which doesn't accept ping(icmp echo), so ping might not be a reliable solution in those case. Please look for an alternative solution(eg: telnet) for those cases.