Test if a port on a remote system is reachable (without telnet)

In the old days, we used `telnet` to see if a port on a remote host was open: `telnet hostname port` would attempt to connect to any port on any host and give you access to the raw TCP stream.

These days, the systems I work on do not have telnet installed (for security reasons), and all outbound connections to all hosts are blocked by default. Over time, it's easy to lose track of which ports are open to which hosts.

Is there another way to test if a port on a remote system is open – using a Linux system with a limited number of packages installed, and `telnet` is not available?

linux networking port

Related: check status of one port on remote host at SO – kenorb Dec 30 '15 at 14:50

11 Answers

Bash has been able to access TCP and UDP ports for a while. From the man page:

```
/dev/tcp/host/port
    If host is a valid hostname or Internet address, and port is an integer port number
    or service name, bash attempts to open a TCP connection to the corresponding socket.
/dev/udp/host/port
    If host is a valid hostname or Internet address, and port is an integer port number
    or service name, bash attempts to open a UDP connection to the corresponding socket.
```

So you could use something like this:

```
xenon-lornix:~> cat < /dev/tcp/127.0.0.1/22
SSH-2.0-OpenSSH_6.2p2 Debian-6
^C pressed here
```

Taa Daa!

edited Jul 31 '15 at 22:04
Peter Mortensen
8,271 16 61 84

answered Jul 22 '13 at 10:37
lornix
8,528 1 24 33

This also seems to work in MinGW. For instance, a remote VNC server on 192.168.2.100 responds with "RFB 003.008" by using "cat < /dev/tcp/192.168.2.100/5900". – Peter Mortensen Jul 21 '15 at 11:26

However on ports that were not open it timed out after 22 seconds (tried on Ubuntu 14.04 (Trusty Tahr) for a remote server). Interestingly, the timeout period is much shorter than the one for nc (see thnee's answer). – Peter Mortensen Jul 21 '15 at 13:00

1 @lornix, ok, but in this case I have to get the same result with use nc without -z option, but it still does not work: # nc -v -w5 127.0.0.1 18080 Connection to 127.0.0.1 18080 port [tcp/*] succeeded! # cat < /dev/tcp/127.0.0.1/18080 Just hangs without any result. Just want to understand when I can use "/dev/tcp/host/port" option – Alexandr Jun 14 '16 at 7:09

4 @Alexandr... actually, "hangs without any result" is pretty much expected behavior. **cat** is waiting for input. **nc** has extra smarts to enable it to sense no-data pending and stops trying. **cat** isn't *quite* as smart. Try `cat < /dev/tcp/localhost/22` , you should get your sshd header. Evidently, your process on port 18080 waits for something to come in, before sending anything. Port 22 (**ssh**) greets you with it's version and whatnot. Try it out! – lornix Jun 14 '16 at 8:44

1 @lornix, thank you very much for explanation! Now the restriction is clear. I think using nc should be a preferred way to check ports. – Alexandr Jun 14 '16 at 14:29

Nice and verbose! From the man pages.
Single port:

```
nc -zv 127.0.0.1 80
```

Multiple ports:

```
nc -zv 127.0.0.1 22 80 8080
```

Range of ports:

```
nc -zv 127.0.0.1 20-30
```

answered Dec 3 '13 at 21:34

Subhranath Chunder
3,261 1 6 2

4 Seems to be far the bestest answer, thanks. ;-) – lpapp Nov 26 '14 at 15:42

4 This hanged when tried on Ubuntu 14.04 (Trusty Tahr) for a remote server (same LAN) for closed ports (it timed out after 127 seconds) - thus not very suitable in scripts. It did work though for a service that had a port open. Using option "-w2" could be the solution. – Peter Mortensen Jul 21 '15 at 12:36

4 Use -u option for UDP ports. – Efren Sep 21 '16 at 0:10

7 On version 6.4 of ncat -z is not recognized. I was able to do without z – smishra Apr 6 '17 at 17:18

3 You can check multiple ranges with: `nc -zv 127.0.0.1 22,80,8080,20-30,443-446` (nc Version: 1.107-4). – bobbel Jul 6 '17 at 16:55

Netcat is a useful tool:

```
nc 127.0.0.1 123 &> /dev/null; echo $?
```

Will output `0` if port 123 is open, and `1` if it's closed.

edited Oct 20 '15 at 18:28

answered Jul 19 '13 at 18:07

thnee
1,451 10 8

This is a far more elegant and scriptable answer than my own. It is unfortunate for me that the security-conscious sysadmins who withheld `telnet` also withheld `nc` (though – strangely – not `curl` or `wget`). – Steve HHH Jul 19 '13 at 19:51

Yes that is completely arbitrary and silly. – thnee Jul 19 '13 at 20:10

3 Let the `FOR` statements begin! – Chad Harrison Jul 19 '13 at 21:37

1 This hanged when tried on Ubuntu 14.04 (Trusty Tahr) for a remote server (same LAN) for closed ports (it timed out after about 127 seconds) - thus not very suitable in scripts. It did work though for a service that had a port open, returning 0. Using option "-w2" could be the solution. – Peter Mortensen Jul 21 '15 at 13:06

I think `-G 2` would be more appropriate for TCP timeout – A B Aug 24 '15 at 21:45

The simplest method, without making use of another tool, such as `socat`, is as described in @lornix's answer above. This is just to add an actual example of how one would make use of the psuedo-device `/dev/tcp/...` within Bash if you wanted to, say, test if another server had a given port accessible via the command line.

Examples

Say I have a host on my network named `skinner`.

```
$ (echo > /dev/tcp/skinner/22) >/dev/null 2>&1 \
    && echo "It's up" || echo "It's down"
It's up

$ (echo > /dev/tcp/skinner/222) >/dev/null 2>&1 && \
    echo "It's up" || echo "It's down"
It's down
```

The reason you want to wrap the `echo > /dev/...` in parentheses like this, `(echo > /dev/...)` is because if you don't, then with tests of connections that are down, you'll get these types of messages showing up.

```
$ (echo > /dev/tcp/skinner/223) && echo hi
bash: connect: Connection refused
bash: /dev/tcp/skinner/223: Connection refused
```

These can't simply be redirected to `/dev/null` since they're coming from the attempt to write out data to the device `/dev/tcp`. So we capture all that output within a sub-command, i.e. `(...cmds...)` and redirect the output of the sub-command.

edited Jul 31 '15 at 22:09 — Peter Mortensen — 8,271 16 61 84

answered Sep 2 '14 at 17:59 — slm — 6,040 5 37 45

This is excellent. Wish it would get voted up to the top. I only read this far down the page because I accidentally scrolled before closing it. – Still.Tony Feb 12 '15 at 13:44

@Okuma.Tony - yes that's always an issue with Q's that have many answers 8-). Thanks for the feedback though, it's appreciated. – slm Feb 12 '15 at 14:26

I found that `curl` may get the job done in a similar way to `telnet`, and `curl` will even tell you which protocol the listener expects.

Construct an HTTP URI from the hostname and port as the first argument to `curl`. If `curl` can connect, it will report a protocol mismatch and exit (if the listener isn't a web service). If `curl` cannot connect, it will time out.

For example, port 5672 on host 10.0.0.99 is either closed or blocked by a firewall:

```
$ curl http://10.0.0.99:5672
curl: (7) couldn't connect to host
```

However, from a different system, port 5672 on host 10.0.0.99 can be reached, and appears to be running an AMQP listener.

```
$ curl http://10.0.0.99:5672
curl: (56) Failure when receiving data from the peer
AMQP
```

It's important to distinguish between the different messages: the first failure was because `curl` could not connect to the port. The second failure is a success test, though `curl` expected an HTTP listener instead of an AMQP listener.

edited Jul 19 '13 at 17:41

answered Jul 19 '13 at 17:05

Steve HHH
2,735 6 21 32

4 If curl isn't available, wget might be. `wget -qS -O- http://ip.add.re.ss:port` should effectively do the same thing. – α CVn Jul 19 '13 at 17:52

2 This even works with a hostname, ex. `curl myhost:22 .` – 에이바 Feb 25 '14 at 16:17

This may be incorrect. I am havng a tomcat service running, but getting 404 error. # curl -k 192.168.194.4:6443 <html><head><title>Apache Tomcat/7.0.34 - Error report</title><style><!--H1 --- HR {color : #525D76;}--></style> </head><body><h1>HTTP Status 404 - /</h1><HR size="1" noshade="noshade"> type Status report message <u>/</u> description <u>The requested resource is not available. </u> <HR size="1" noshade="noshade"><h3>Apache Tomcat/7.0.34</h3></body></html> – Mohammad Shahid Siddiqui Jun 2 '15 at 6:30

See my post with similar approach. – kenorb Dec 30 '15 at 14:49

```
[admin@automation-server 1.2.2]# nc -v -z -w2 192.168.193.173 6443
nc: connect to 192.168.193.173 port 6443 (tcp) failed: Connection refused

[admin@automation-server 1.2.2]# nc -v -z -w2 192.168.194.4 6443
Connection to 192.168.194.4 6443 port [tcp/sun-sr-https] succeeded!
```

Hope it solves your problem :)

answered Jun 2 '15 at 6:27

Mohammad Shahid Siddiqui
201 2 2

1 Yes, this is better - timing out almost immediately for closed ports. – Peter Mortensen Jul 21 '15 at 12:38

Does this always use TCP or is there a way to get it to check UDP? – kmoe Jan 2 '16 at 17:09

Here is one-liner:

```
</dev/tcp/localhost/11211 && echo Port is open || echo Port is closed
```

using Bash syntax explained in @lornix answer.

For more info, check: Advanced Bash-Scripting Guide: Chapter 29. `/dev` and `/proc` .

edited Jul 30 at 10:37

answered Mar 16 '16 at 11:21

kenorb


10.2k 15 74 106

I was struggling for a whole day because none of these answers seemed to work for me. The problem is that the most recent version of `nc` no longer has the `-z` flag, whereas direct access via TCP (as according to @lornix and @slm) fails when the host is not reachable. I eventually found this page, where I finally found not one but *two* working examples:

1. `nc -w1 127.0.0.1 22 </dev/null`

   (the `-w` flag takes care of the timeout, and the `</dev/null` replaces the `-z` flag)

2. `timeout 1 bash -c '(echo > /dev/tcp/127.0.0.1/22) >/dev/null 2>&1'`

   (the `timeout` command takes care of the timeout, and the rest is from @slm)

Then, simply use `&&` and/or `||` (or even `$?`) to extract the result. Hopefully, somebody will find this information useful.

answered May 31 '17 at 8:45


Azukikuru
61 1 1

It shouldn't be available on your box, but try with `nmap`.

edited Jul 21 '15 at 11:28


Peter Mortensen
8,271 16 61 84

answered Jul 19 '13 at 17:01


peperunas
426 2 8 15

4 `nmap` is a good tool, but not available on these systems. Rather than download `nmap`, compile it, install it to my home directory, then copy it to all the other systems, I was hoping to find a way using existing tools available in most Linux installations. – Steve HHH Jul 19 '13 at 17:16

Combining the answers from @kenorb and @Azukikuru you could test port open/closed/firewalled.

```
timeout 1 bash -c '</dev/tcp/127.0.0.1/22 && echo Port is open || echo Port is closed' ||
echo Connection timeout
```

Another approach with curl for reaching any port

```
curl telnet://127.0.0.1:22
```

edited Aug 12 at 16:54

answered Aug 10 at 12:20


Miguel Ferreira
101 2

If you've to test more than on system you may use our test tool dda-serverspec (https://github.com/DomainDrivenArchitecture/dda-serverspec-crate) for such tasks. You may define your expectation

```
{:netcat [{:host "mywebserver.com" :port "443"}
          {:host "telnet mywebserver.com" :port "80"}
```

```
           {:host "telnet mywebserver.com" :port "8443"}]}
```

and test these expectation either against localhost or against remote hosts (connect by ssh). For remote tests you've to define a targets:

```
{:existing [{:node-name "test-vm1"
            :node-ip "35.157.19.218"}
           {:node-name "test-vm2"
            :node-ip "18.194.113.138"}]
 :provisioning-user {:login "ubuntu"}}
```

You may run the test with `java -jar dda-serverspec.jar --targets targets.edn serverspec.edn`

Under the hood we're using netcat as proprosed above ...

answered Oct 26 at 13:49

jerger
1

New contributor