iperf3 Network throughput tester

Server iperf3 -s, port 5201 — control → — **Client** iperf3 -c host — data streams → — **Interval reports** every 1 s — summary → — **Sender / receiver** final totals

CONCEPTS

iperf3 needs a server at one end and a client at the other; it measures the path between them, not either machine alone. Test only networks and hosts you are authorised to use.

Default direction is client to server. -R reverses it, so the client receives; --bidir does both at once. Open TCP and UDP port 5201 in any firewall in between.

SERVER

`iperf3 -s`
Listen for tests (port 5201)

`iperf3 -s -p <port>`
Listen on another port

`iperf3 -s -1`
Serve one client, then exit

`iperf3 -s -D`
Run as a background daemon

`iperf3 -s -D --logfile <file>`
Daemon that logs to a file

`iperf3 -s -B <addr>`
Bind to one local address

`iperf3 -s -I <file>`
Write the PID to a file

RUNNING A TEST

`iperf3 -c <host>`
Run a TCP test

`iperf3 -c <host> -p <port>`
Connect to another port

`iperf3 -c <host> -t <sec>`
Set duration (default 10 s)

`iperf3 -c <host> -n <bytes>`
Send a fixed amount, e.g. 2G

`iperf3 -c <host> -O <sec>`
Skip the first seconds in stats

`iperf3 -c <host> -4 / -6`
Force IPv4 or IPv6

`iperf3 -c <host> -B <addr>`
Bind to a local address

DIRECTION & STREAMS

`iperf3 -c <host> -R`
Reverse: server sends to client

`iperf3 -c <host> --bidir`
Send and receive at the same time

`iperf3 -c <host> -P <n>`
Run n parallel streams

DIRECTION & STREAMS (cont.)

`iperf3 -c <host> -P <n> -R`
Parallel streams, reversed

`iperf3 -c <host> -A <n>`
Pin to a CPU core

UDP & BITRATE

`iperf3 -c <host> -u`
Run a UDP test (1 Mbit/s default)

`iperf3 -c <host> -u -b <rate>`
UDP at a target rate, e.g. 100M

`iperf3 -c <host> -u -b 0`
UDP as fast as possible

`iperf3 -c <host> -b <rate>`
Cap a TCP test's rate

`iperf3 -c <host> -u -l <bytes>`
Set the datagram size

`iperf3 -c <host> -u -R`
UDP loss and jitter, download

TUNING

`iperf3 -c <host> -w <size>`
Set socket buffer or window

TUNING (cont.)

`iperf3 -c <host> -M <mss>`
Set TCP maximum segment size

`iperf3 -c <host> -N`
Disable Nagle's algorithm

`iperf3 -c <host> -C <algo>`
Pick a congestion control (e.g. bbr)

`iperf3 -c <host> -l <size>`
Set the read/write buffer length

OUTPUT

`iperf3 -c <host> -J`
Output as JSON

`iperf3 -c <host> -i <sec>`
Report interval (0 turns it off)

`iperf3 -c <host> -f <unit>`
Units: k, m, g, t (bits) or K, M, G, T (bytes)

`iperf3 -c <host> -V`
More detailed output

`iperf3 -c <host> --get-server-output`
Also print the server's view

OUTPUT (cont.)

`iperf3 -c <host> --logfile <file>`
Write output to a file

READING RESULTS

Output columns

Column	Meaning
`Transfer`	Data moved in the interval
`Bitrate`	Throughput in that interval
`Retr`	TCP retransmits (sender)
`Cwnd`	Congestion window (sender)
`Jitter`	UDP delay variation
`Lost/Total`	UDP datagrams lost / sent

Many Retr values point to loss or congestion on the path. For UDP, loss at a low rate means a real problem; loss only near the link rate means you just filled it. Read the receiver line for the true figure.

A single run is a sample. Repeat it, vary -P and -R, and compare TCP with UDP before blaming the network, since CPU, NIC offloads and Wi-Fi can all limit the result.