aria2 Fast multi-download

START HERE

aria2c *<url>*
Download a file

aria2c -d *<dir>* **-o** *<name>* *<url>*
Choose folder and name

aria2c -x 8 -s 8 *<url>*
Use 8 connections

aria2c "*<magnet>***"**
Fetch a magnet link

aria2c -c *<url>*
Resume a download

aria2c *<file>***.torrent**
Fetch a torrent

aria2c -i *<file>*
Download a URL list

aria2c --max-download-limit=1M *<url>*
Cap the speed

Source URL, magnet, .torrent → **-s splits** → **Pieces** min size -k → **-x connections** → **Parallel streams** per server / peers → **writes** → **File** -d dir, -o name

CONCEPTS

aria2c takes URLs, magnets, .torrent and .metalink files, and mixes protocols for one file.

-x limits connections per server (max 16); -s is how many pieces to split into. Raise both for more speed.

Only download content you are allowed to copy; respect copyright and the site's terms.

aria2c exits with status 0 only if every download succeeded, so scripts can test it.

DOWNLOADING

aria2c *<url>*
Download one file to the current dir

aria2c *<url1> <url2>*
Download several files

aria2c *<url> <mirror-url>*
Same file from several mirrors

aria2c -d *<dir> <url>*
Save into a directory

aria2c -o *<name> <url>*
Save under another name

aria2c -c *<url>*
Continue a partial download

aria2c -x *<n> <url>*
Connections per server
The default is 1; 16 is the maximum.

aria2c -s *<n> <url>*
Split the file into n parts

aria2c -k *<size> <url>*
Minimum split size, e.g. 1M

aria2c -Z *<url1> <url2>*
Treat each URL as a separate file
Needed when giving mirrors that should not be merged into one download.

aria2c --allow-overwrite=true *<url>*
Replace an existing file

aria2c --auto-file-renaming=false *<url>*
Never rename on name clash

aria2c --conditional-get=true *<url>*
Skip if the local file is current

DOWNLOADING (cont.)

aria2c --checksum=*<type>***=***<digest>* *<url>*
Verify the download
Example type: sha-256.

LISTS AND QUEUE

aria2c -i *<file>*
Read URLs from a file

aria2c -j *<n>* **-i** *<file>*
Run n downloads at once

aria2c --save-session=*<file>* **-i** *<file>*
Save unfinished downloads on exit

aria2c --dry-run=true *<url>*
Check availability without downloading

aria2c --max-tries=*<n>* **--retry-wait=***<sec>*
Retry count and pause

aria2c -t *<sec>*
Connection timeout

aria2c --lowest-speed-limit=*<speed>*
Drop connections slower than this

aria2c --on-download-complete=*<cmd>*
Run a command when done

aria2c --file-allocation=*<method>*
none, prealloc, trunc or falloc

URL list with per-file options

```
https://example.com/a.iso
  dir=/data/isos
  out=a-renamed.iso
# mirrors: put them on one line, TAB-separated
```

SPEED AND LIMITS

aria2c --max-download-limit=*<speed>* *<url>*
Cap one download, e.g. 500K

aria2c --max-overall-download-limit=*<speed>*
Cap all downloads together

aria2c --max-upload-limit=*<speed>*
Cap upload per torrent

aria2c --max-overall-upload-limit=*<speed>*
Cap total upload

SPEED AND LIMITS (cont.)

aria2c --summary-interval=*<sec>*
How often to print the summary

aria2c -l *<file>*
Write a log file

aria2c --console-log-level=*<level>*
debug, info, notice, warn, error

aria2c -D *<url>*
Run in the background

TORRENTS AND MAGNETS

aria2c "*<magnet-uri>***"**
Download a magnet link
Quote it: the shell would eat the & characters.

aria2c *<file>***.torrent**
Download a torrent file

aria2c -S *<file>***.torrent**
List files in a torrent

aria2c --select-file=*<n,m-k>* *<file>***.torrent**
Only these file indexes

aria2c --seed-time=*<min>* *<file>***.torrent**
Seed this many minutes after done

aria2c --seed-ratio=*<ratio>* *<file>***.torrent**
Seed until this share ratio
Default is 1.0; 0.0 seeds forever when no seed-time is set.

aria2c --seed-time=0 *<file>***.torrent**
Do not seed

aria2c -T *<file>***.torrent** *<url>*
Torrent plus HTTP web seed

aria2c --bt-tracker=*<url,url>*
Add extra trackers

aria2c --enable-dht=false
Turn DHT off

aria2c --listen-port=*<port>*
Port for incoming BitTorrent

aria2c --follow-torrent=false *<file>***.torrent**
Do not open the .torrent it fetched

aria2c -M *<file>***.metalink**
Download from a metalink

HTTP AND AUTH

aria2c --header="*<Name: value>***"** *<url>*
Add a request header

aria2c -U *<agent> <url>*
Set the User-Agent

aria2c --http-user=*<u>* **--http-passwd=***<p> <url>*
HTTP basic auth

aria2c --load-cookies=*<file> <url>*
Send cookies from a file

aria2c --all-proxy=*<host:port> <url>*
Use a proxy for all protocols

aria2c --referer=*<url>* *<url>*
Set the Referer

aria2c --check-certificate=false *<url>*
Skip TLS verification
Unsafe: only for hosts you trust.

aria2c -V *<file>*
Re-check integrity of existing data

RPC DAEMON

aria2c --enable-rpc
Start the JSON-RPC server

aria2c --enable-rpc --rpc-listen-port=*<port>*
Change RPC port (6800)

aria2c --rpc-secret=*<token>*
Require a token for RPC calls

Web UIs such as AriaNg talk to the RPC server; always set a secret before exposing the port.

RECIPES

Fast resumable download

```
aria2c -c -x 16 -s 16 -k 1M \
  -d ~/Downloads <url>
```

Batch from a list, 3 at a time

```
aria2c -c -j 3 -x 4 -i urls.txt \
  -d downloads --save-session=left.txt
```

Fetch one file from a torrent

```
aria2c -S debian.torrent
aria2c --select-file=2 --seed-time=0 debian.torrent
```

Authenticated, limited download

```
aria2c --header="Authorization: Bearer <token>" \
  --max-download-limit=2M -o data.zip <url>
```