ArchiveBot Documentation

Release 1.6

ArchiveTeam

Jan 04, 2018

Contents

1 Commands **3**
- 1.1 archive . . . 3
- 1.2 abort . . . 5
- 1.3 archiveonly . . . 5
- 1.4 explain . . . 7
- 1.5 archiveonly < FILE . . . 7
- 1.6 ignore . . . 7
- 1.7 unignore . . . 8
- 1.8 ignoreset . . . 8
- 1.9 ignorereports . . . 9
- 1.10 delay . . . 9
- 1.11 concurrency . . . 9
- 1.12 yahoo . . . 9
- 1.13 expire . . . 10
- 1.14 status . . . 10
- 1.15 pending . . . 11
- 1.16 whereis . . . 11

2 ArchiveBot Redis layout **13**
- 2.1 Connection . . . 13
- 2.2 `pipeline:PIPELINE_ID` . . . 13
- 2.3 Hash keys . . . 14
- 2.4 `IDENT` (i.e. `[a-z0-9]{25,}`) . . . 14
- 2.5 Hash keys . . . 14
- 2.6 `IDENT_ignores` . . . 15
- 2.7 `IDENT_log` . . . 15
- 2.8 `pipelines` . . . 16
- 2.9 `jobs_completed`, `jobs_aborted`, `jobs_failed` . . . 16
- 2.10 `tweets:done`, `tweets:queue` . . . 16
- 2.11 Pubsub channels . . . 16
- 2.12 `updates` . . . 16
- 2.13 `archivebot:job:IDENT` . . . 16

3 ArchiveBot Administration **17**
- 3.1 Basic Information . . . 17
- 3.2 How to add new ArchiveBot pipelines . . . 18
- 3.3 All about tmux . . . 18

3.4 Re-sync the IRC !status command to actual Redis data 21

4 Indices and tables **23**

Homepage http://www.archiveteam.org/index.php?title=ArchiveBot

Contents:

CHAPTER 1

Commands

ArchiveBot listens to commands prefixed with `!`.

1.1 archive

`!archive URL`, `!a URL` begin recursive retrieval from a URL

```
> !archive http://artscene.textfiles.com/litpacks/
< Archiving http://artscene.textfiles.com/litpacks/.
< Use !status 43z7a11vo6of3a7i173441dtc for updates, !abort
  43z7a11vo6of3a7i173441dtc to abort.
```

ArchiveBot does not ascend to parent links. This means that everything under the `litpacks` directory will be downloaded. For example, `/litpacks/hello.html` will be downloaded but not `/hello.html`.

If you leave out the trailing slash, eg `/litpacks`, it will consider that to be a file and download everything under `/`.

URLs are treated as case-sensitive. `/litpacks` is different from `/LitPacks`.

1.1.1 Accepted parameters

`--ignore-sets SET1,...,SETN` specify sets of URL patterns to ignore:

```
> !archive http://example.blogspot.com/ncr --ignore-sets=blogs,forums
< Archiving http://example.blogspot.com/ncr.
< 14 ignore patterns loaded.
< Use !status 5sid4pgxkiu6zynhbt3q1gi2s for updates, !abort
  5sid4pgxkiu6zynhbt3q1gi2s to abort.
```

Known sets are listed in db/ignore_patterns/.

Aliases: `--ignoresets`, `--ignore_sets`, `--ignoreset`, `--ignore-set`, `--ignore_set`, `--ig-set`, `--igset`

--no-offsite-links do not download links to offsite pages:

```
> !archive http://example.blogspot.com/ncr
>    --no-offsite-links
< Archiving http://example.blogspot.com/ncr.
< Offsite links will not be grabbed.
< Use !status 5sid4pgxkiu6zynhbt3q1gi2s for updates, !abort
  5sid4pgxkiu6zynhbt3q1gi2s to abort.
```

ArchiveBot's default behavior with !archive is to recursively fetch all pages that are descendants of the starting URL, as well as all linked pages and their requisites. This is often useful for preserving a page's context in time. However, this can sometimes result in an undesirably large archive. Specifying `--no-offsite-links` preserves recursive retrieval but does not follow links to offsite hosts.

Please note that ArchiveBot considers `www.example.com` and `example.com` to be different hosts, so if you have a website that uses both, you should not specify `--no-offsite-links`.

Aliases: `--nooffsitelinks`, `--no-offsite`, `--nooffsite`

--user-agent-alias ALIAS specify a user-agent to use:

```
> !archive http://artscene.textfiles.com/litpacks/
    --user-agent-alias=firefox
< Archiving http://artscene.textfiles.com/litpacks/.
< Using user-agent Mozilla/5.0 (Windows NT 5.1; rv:31.0)
  Gecko/20100101 Firefox/31.0.
< Use !status 43z7a11vo6of3a7i173441dtc for updates, !abort
  43z7a11vo6of3a7i173441dtc to abort.
```

This option makes the job present the given user-agent. It can be useful for archiving sites that (still) do user-agent detection.

See db/user_agents for a list of recognized aliases.

Aliases: `--useragentalias`, `--user-agent`, `--useragent`

--pipeline TAG specify which pipeline to use:

```
> !archive http://example.blogspot.com/ncr
    --pipeline=superfast
< Archiving http://example.blogspot.com/ncr.
< Job will run on a pipeline whose name contains "superfast".
< Use !status 5sid4pgxkiu6zynhbt3q1gi2s for updates, !abort
  5sid4pgxkiu6zynhbt3q1gi2s to abort.
```

Pipeline operators assign nicknames to pipelines. Oftentimes, these nicknames describe the pipeline: datacenter, special modifications, etc. This option can be used to load jobs onto those pipelines.

In the above example, both of the following pipeline nicks would match the given tag:

- superfast
- ovhca1-superfast-47

NOTE: You should use a pipeline *nickname* for this command, not one of the auto-assigned pipeline *id numbers* like `1a5adaacbe686c708f9277e7b70b590c`.

--phantomjs access pages via PhantomJS

--phantomjs-wait set number of seconds between PhantomJS requests; defaults to 2.0

--phantomjs-scroll maximum number of times to scroll a page in PhantomJS; defaults to 100

--no-phantomjs-smart-scroll disable PhantomJS' end-of-page detection and always scroll `--phantomjs-scroll` number of times; off by default

PhantomJS mode is enabled if any of the `--*phantomjs*` options are passed.

--explain alias for `!explain` adds a short note explaining the purpose of the archiving job

--delay alias for `!delay` (in milliseconds) only allows a single value; to provide a range, use `!delay`

--concurrency alias for `!concurrency` sets number of workers for job (use with care!)

1.2 abort

!abort IDENT abort a job:

```
> !abort 1q2qydhkeh3gfnrcxuf6py70b
< Initiating abort for job 1q2qydhkeh3gfnrcxuf6py70b.
```

At the moment, a job is not actually aborted and removed from the `!pending` job queue until all the jobs in front of it have started.

1.3 archiveonly

!archiveonly URL, **!ao URL** non-recursive retrieval of the given URL:

```
> !archiveonly http://store.steampowered.com/livingroom
< Archiving http://store.steampowered.com/livingroom without
  recursion.
> Use !status 1q2qydhkeh3gfnrcxuf6py70b for updates, !abort
  1q2qydhkeh3gfnrcxuf6py70b to abort.
```

1.3.1 Accepted parameters

--ignore-sets SET1,...,SETN specify sets of URL patterns to ignore:

```
> !archiveonly http://example.blogspot.com/ --ignore-sets=blogs,forums
< Archiving http://example.blogspot.com/ without recursion.
< 14 ignore patterns loaded.
< Use !status 5sid4pgxkiu6zynhbt3q1gi2s for updates, !abort
  5sid4pgxkiu6zynhbt3q1gi2s to abort.
```

Known sets are listed in db/ignore_patterns/.

--user-agent-alias ALIAS specify a user-agent to use:

```
> !archiveonly http://artscene.textfiles.com/litpacks/
    --user-agent-alias=firefox
< Archiving http://artscene.textfiles.com/litpacks/ without
  recursion.
< Using user-agent Mozilla/5.0 (Windows NT 5.1; rv:31.0)
  Gecko/20100101 Firefox/31.0.
< Use !status 43z7a11vo6of3a7i173441dtc for updates, !abort
  43z7a11vo6of3a7i173441dtc to abort.
```

This option makes the job present the given user-agent. It can be useful for archiving sites that (still) do user-agent detection. See db/user_agents for a list of recognized aliases.

--pipeline TAG specify pipeline to use:

```
> !archiveonly http://example.blogspot.com/
    --pipeline=superfast
< Archiving http://example.blogspot.com/.
< Job will run on a pipeline whose name contains "superfast".
< Use !status 5sid4pgxkiu6zynhbt3q1gi2s for updates, !abort
  5sid4pgxkiu6zynhbt3q1gi2s to abort.
```

--youtube-dl

Warning: This is an often-glitchy feature and not all pipelines support it. To find a pipeline that supports youtube-dl, use the ArchiveBot pipeline monitor page and look for a pipeline whose version is newer than 20150512.01. Also note that this command will only work when using `!archiveonly` or `!ao` to crawl specific individual web pages with embedded video, and this will not work recursively on an entire `!archive` or `!a` website grab.

Attempt to download videos using youtube-dl (experimental):

```
> !archiveonly https://example.website/fun-video-38214 --youtube-dl
< Queued https://example.website/fun-video-38214 for archival without
  recursion.
< Options: youtube-dl: yes
< Use !status dma5g7xcy0r3gbmisqshkpkoe for updates, !abort
  dma5g7xcy0r3gbmisqshkpkoe to abort.
```

When –youtube-dl is passed, ArchiveBot will attempt to download videos embedded in HTML pages it encounters in the crawl using youtube-dl (http://rg3.github.io/youtube-dl/). youtube-dl can recognize many different embedding formats, but success is not guaranteed.

If you are going to use this option, please watch your job's progress on the dashboard. If you see MP4 or WebM files in the download log, your videos were probably saved. (You can click on links in the download log to confirm.)

Video playback is not yet well-supported in web archive playback tools. As of May 2015:

- pywb v0.9 (https://github.com/ikreymer/pywb) is known to work.
- https://github.com/ikreymer/webarchiveplayer is based on pywb 0.8, and might work.
- The Internet Archive's Wayback Machine does not present videos in ArchiveBot WARCs. (Wayback may not support the record convention used by ArchiveBot and/or may not support video playback at all.)

--phantomjs access pages via PhantomJS

--phantomjs-wait set number of seconds between PhantomJS requests; defaults to 2.0

--phantomjs-scroll maximum number of times to scroll a page in PhantomJS; defaults to 100

--no-phantomjs-smart-scroll disable PhantomJS' end-of-page detection and always scroll `--phantomjs-scroll` number of times; off by default

PhantomJS mode is enabled if any of the `--*phantomjs*` options are passed.

1.4 explain

`!explain IDENT NOTE`, **`!ex IDENT NOTE`** add a short note to explain why this site is being archived:

```
> !explain byu50bzfdbnlyl6mrgn6dd24h shutting down 7/31
> Added note "shutting down 7/31" to job byu50bzfdbnlyl6mrgn6dd24h.
```

Pipeline operators (really, anyone) may want to know why a job is running. This becomes particularly important when a job grows very large (hundreds of gigabytes). While this can be done via IRC, IRC communication is asynchronous, people can be impatient, and a rationale can usually be summed up very concisely.

1.5 archiveonly < FILE

`!archiveonly < URL`, **`!ao < URL`** archive each URL in the text file at URL:

```
> !archiveonly < https://www.example.com/some-file.txt
< Archiving URLs in https://www.example.com/some-file.txt without
  recursion.
> Use !status byu50bzfdbnlyl6mrgn6dd24h for updates, !abort
  byu50bzfdbnlyl6mrgn6dd24h to abort.
```

The text file should list one URL per line. Both UNIX and Windows line endings are accepted.

1.5.1 Accepted parameters

`!archiveonly < URL` accepts the same parameters as `!archiveonly`. A quick reference:

`--ignore-sets SET1,...,SETN` specify sets of URL patterns to ignore

`--user-agent-alias ALIAS` specify a user-agent to use

`--pipeline TAG` specify pipeline to use

`--youtube-dl` attempt to download videos using youtube-dl

`--phantomjs` access pages via PhantomJS

`--phantomjs-wait` set number of seconds between PhantomJS requests; defaults to 2.0

`--phantomjs-scroll` maximum number of times to scroll a page in PhantomJS; defaults to 100

`--no-phantomjs-smart-scroll` disable PhantomJS' end-of-page detection and always scroll `--phantomjs-scroll` number of times; off by default

1.6 ignore

`!ignore IDENT PATTERN`, **`!ig IDENT PATTERN`** add an ignore pattern:

```
> !ig 1q2qydhkeh3gfnrcxuf6py70b obnoxious\?foo=\d+
< Added ignore pattern obnoxious\?foo=\d+ to job
  1q2qydhkeh3gfnrcxuf6py70b.
```

The pattern must be expressed as regular expressions. For more information, see:

- http://docs.python.org/3/howto/regex.html#regex-howto

- http://docs.python.org/3/library/re.html#regular-expression-syntax

Two strings, `{primary_url}` and `{primary_netloc}`, have special meaning.

`{primary_url}` expands to the top-level URL. For `!archive` jobs, this is the initial URL. For `!archiveonly < FILE` jobs, `{primary_url}` is the top-level URL that owns the descendant being archived.

`{primary_netloc}` is the auth/host/port section of `{primary_url}`.

1.6.1 Examples

1. To ignore everything on domain1.com and its subdomains, use pattern `^https?://([^/]+\.)?domain1\.com/`
2. To ignore everything *except* URLs on domain1.com or domain2.com, use pattern `^(?!https?://(domain1\.com|domain2\.com)/)`
3. To keep subdomains on domain1.com as well, use pattern `^(?!https?://(([^/]+\.)?domain1\.com|domain2\.com)/)`
4. For `!archive` jobs on subdomain blogs (such as Tumblr), the following pattern ignores all URLs except the initial URL, sub-URLs of the initial URL, and media/asset servers: `^http://(?!({primary_netloc}|\d+\.media\.example\.com|assets\.example\.com)).*`
5. Say you have this URL file:

   ```
   http://www.example.com/foo.html
   http://www.bar.org:8080/qux.html
   ```

   and you submit it as an `!archiveonly < FILE` job.

   When retrieving requisites of `http://www.example.com/foo.html`, `{primary_url}` will be `http://www.example.com/foo.html` and `{primary_netloc}` will be `www.example.com`.

   When retriving requisites of `http://www.bar.org:8080/qux.html`\`, `{primary_url}` will be `http://www.bar.org:8080/qux.html` and `{primary_netloc}` will be `www.bar.org:8080`.

1.7 unignore

`!unignore IDENT PATTERN`, `!unig IDENT PATTERN`, `!ug IDENT PATTERN` remove an ignore pattern:

```
> !unig 1q2qydhkeh3gfnrcxuf6py70b obnoxious\?foo=\d+
< Removed ignore pattern obnoxious\?foo=\d+ from job
  1q2qydhkeh3gfnrcxuf6py70b.
```

1.8 ignoreset

`!ignoreset IDENT NAME`, `!igset IDENT NAME` add a set of ignore patterns:

```
> !igset 1q2qydhkeh3gfnrcxuf6py70b blogs
< Added 17 ignore patterns to job 1q2qydhkeh3gfnrcxuf6py70b.
```

You may specify multiple ignore sets. Ignore sets that are unknown are, well, ignored:

```
> !igset 1q2qydhkeh3gfnrcxuf6py70b blogs, other
< Added 17 ignore patterns to job 1q2qydhkeh3gfnrcxuf6py70b.
< The following sets are unknown: other
```

Ignore set definitions can be found under db/ignore_patterns/.

1.9 ignorereports

`!ignorereports IDENT on|off`, **`!igrep IDENT on|off`** toggle ignore reports:

```
> !igrep 1q2qydhkeh3gfnrcxuf6py70b on
< Showing ignore pattern reports for job 1q2qydhkeh3gfnrcxuf6py70b.

> !igrep 1q2qydhkeh3gfnrcxuf6py70b off
< Suppressing ignore pattern reports for job
  1q2qydhkeh3gfnrcxuf6py70b.
```

Some jobs generate ignore patterns at high speed. For these jobs, turning off ignore pattern reports may improve both the usefulness of the dashboard job log and the speed of the job.

This command is aliased as `!igoff IDENT` and `!igon IDENT`. !igoff suppresses reports; !igon shows reports.

1.10 delay

`!delay IDENT MIN MAX`, **`!d IDENT MIN MAX`** set inter-request delay:

```
> !delay 1q2qydhkeh3gfnrcxuf6py70b 500 750
< Inter-request delay for job 1q2qydhkeh3gfnrcxuf6py70b set to [500,
  750 ms].
```

Delays may be any non-negative number, and are interpreted as milliseconds. The default inter-request delay range is [250, 375] ms.

1.11 concurrency

`!concurrency IDENT LEVEL`, **`!con IDENT LEVEL`** set concurrency level:

```
> !concurrency 1q2qydhkeh3gfnrcxuf6py70b 8
< Job 1q2qydhkeh3gfnrcxuf6py70b set to use 8 workers.
```

Adding additional workers may speed up grabs if the target site has capacity to spare, but it also puts additional pressure on the target. Use wisely.

1.12 yahoo

`!yahoo IDENT` set zero second delays, crank concurrency to 4:

```
> !yahoo 1q2qydhkeh3gfnrcxuf6py70b
< Inter-request delay for job 1q2qydhkeh3gfnrcxuf6py70b set to
  [0, 0] ms.
< Job 1q2qydhkeh3gfnrcxuf6py70b set to use 4 workers.
```

Only recommended for use when archiving data from hosts with gobs of bandwidth and processing power (e.g. Yahoo, Google, Amazon). Keep in mind that this is likely to trigger any rate limiters that the target may have.

1.13 expire

`!expire IDENT` for expiring jobs, expire a job immediately:

```
> !expire 1q2qydhkeh3gfnrcxuf6py70b
< Job 1q2qydhkeh3gfnrcxuf6py70b expired.
```

In rare cases, the 48 hour timeout enforced by ArchiveBot on archive jobs is too long. This command permits faster snapshotting. It should be used sparingly, and only ops are able to use it; abuse is very easy to spot.

If a job's expiry timer has not yet started, this command does not affect the given job:

```
> !expire 5sid4pgxkiu6zynhbt3q1gi2s
< Job 5sid4pgxkiu6zynhbt3q1gi2s does not yet have an expiry timer.
```

This is intended to prevent expiration of active jobs.

1.14 status

`!status` print job summary:

```
> !status
< Job status: 0 completed, 0 aborted, 0 in progress, 0 pending
```

`!status IDENT`, **`!status URL`** print information about a job or URL

For an unknown job:

```
> !status 1q2qydhkeh3gfnrcxuf6py70b
< Sorry, I don't know anything about job 1q2qydhkeh3gfnrcxuf6py70b.
```

For a URL that hasn't been archived:

```
> !status http://artscene.textfiles.com/litpacks/
< http://artscene.textfiles.com/litpacks/ has not been archived.
```

For a URL that hasn't been archived, but has children that have been processed before (either succesfully or unsuccessfully):

```
> !status http://artscene.textfiles.com/
< http://artscene.textfiles.com/ has not been archived.
< However, there have been 5 download attempts on child URLs.
< More info: http://www.example.com/#/prefixes/http://artscene.textfiles.com/
```

For an ident or URL that's in progress:

```
> !status 43z7a11vo6of3a7i173441dtc
<
< Downloaded 10.01 MB, 2 errors encountered
< More info at my dashboard: http://www.example.com
```

For an ident or URL that has been successfully archived within the past 48 hours:

```
> !status 43z7a11vo6of3a7i173441dtc
< Archived to http://www.example.com/site.warc.gz
< Eligible for rearchival in 30h 25m 07s
```

For an ident or URL identifying a job that was aborted:

```
> !status 43z7a11vo6of3a7i173441dtc
< Job aborted
< Eligible for rearchival in 00h 00m 45s
```

1.15 pending

`!pending` send pending queue in private message:

```
> !pending
< [privmsg] 2 pending jobs:
< [privmsg] 1. http://artscene.textfiles.com/litpacks/
              (43z7a11vo6of3a7i173441dtc)
< [privmsg] 2. http://example.blogspot.com/ncr
              (5sid4pgxkiu6zynhbt3q1gi2s)
```

Jobs are listed in the order that they'll be worked on. This command lists only the global queue; it doesn't yet show the status of any pipeline-specific queues.

1.16 whereis

`!whereis IDENT`, **`!w IDENT`** display which pipeline the given job is running on:

```
> !whereis 1q2qydhkeh3gfnrcxuf6py70b
< Job 1q2qydhkeh3gfnrcxuf6py70b is on pipeline
  "pipeline-foobar-1" (pipeline:abcdef1234567890).
```

For jobs not yet on a pipeline:

```
> !status 43z7a11vo6of3a7i173441dtc
< Job 43z7a11vo6of3a7i173441dtc is not on a pipeline.
```

CHAPTER 2

ArchiveBot Redis layout

The ArchiveBot pipelines and backend share a single Redis database. This document describes the keys in that database.

Keys do not follow any namespace-prefixing convention; ArchiveBot assumes it has full control over the database.

2.1 Connection

Pipelines connect directly to the Redis database, typically over SSH or spiped. The backend connects the same way. There is no access control from either side.

2.2 `pipeline:PIPELINE_ID`

Type: hash

Keys matching this form describe pipelines. PIPELINE_ID is a hexadecimal number that is generated by a pipeline process on startup. The pipeline process periodically updates its data while it runs.

2.3 Hash keys

Key	Intended type	Meaning
disk_usage	Decimal	% of the pipeline's filesystem in use
disk_available	Integer	Bytes available on the pipeline's filesystem
fqdn	String	FQDN of the host running the pipeline
hostname	String	Short name of the host
id	String	The pipeline's ID; always matches the hash key
load_average_1m	Decimal	Load average over the past minute
load_average_5m	Decimal	"" "" "" "" over the past 5 minutes
load_average_15m	Decimal	"" "" "" "" over the past 15 minutes
mem_available	Integer	Bytes of memory available on the host
mem_usage	Decimal	% memory in use on the host
nickname	String	The pipeline nickname
pid	Integer	The PID of the pipeline process
python	String	The version of Python running the pipeline
ts	UNIX timestamp	The last time this pipeline record was updated
version	String	The pipeline's version

2.4 `IDENT` (i.e. `[a-z0-9]{25,}`)

Type: hash

These are job records. These are the most common record type in ArchiveBot's database.

Parts of this record are frequently modified by both the backend and pipeline:

- whenever a response is recorded
- whenever job settings are changed

2.5 Hash keys

Key	Intended type	Meaning
bytes_downloaded	Integer	Bytes downloaded from the target site
concurrency	Integer	Current number of concurrent downloaders
death_timer	Integer	Number of liveness checks that have gone without a response
delay_max	Integer	Maximum delay between two requests on a downloader in ms
delay_min	Integer	Minimum delay between two requests on a downloader in ms
error_count	Integer	Number of error (i.e. 4xx, 5xx) responses encountered
fetch_depth	String	"shallow" for !ao jobs; "inf" for !a jobs
finished_at	UNIX ts w/ frac	When the job finished; not present if the job is running
heartbeat	Integer	Set by the pipeline; incremented once per heartbeat
grabber	String	"phantomjs" for PhantomJS jobs; omitted otherwise
ignore_patterns_set_key	String	The key storing this job's ignore patterns
items_downloaded	Integer	Number of 2xx/3xx responses
items_queued	Integer	Number of URLs encountered in the job
last_acknowledged_heartbeat	Integer	Set by the backend; is the last heartbeat received

Continued on next page

Table 2.1 – continued from previous page

Key	Intended type	Meaning
last_analyzed_log_entry	Integer	The last log entry index analyzed by the backend [1]
last_broadcasted_log_entry	Integer	“” “” “” “” “” “” “” “” broadcasted over the firehose [1]
last_trimmed_log_entry	Integer	“” “” “” “” “” “” “” “” trimmed by the log trimmer [1]
log_key	String	The key storing this job’s log messages
log_score	Integer	The current log entry index
next_watermark	Integer	A threshold for number of queued URLs; currently unused
no_phantomjs_smart_scroll	Boolean	Whether or not PhantomJS’ smart-scroll should be used
phantomjs_scroll	Integer	Maximum number of times to scroll the page with PhantomJS
phantomjs_wait	Integer	Maximum wait time between PhantomJS page interactions
pipeline_id	String	The pipeline running this job; corresponds to a pipeline:* key
queued_at	UNIX ts w/ frac	When this job was queued
r1xx	Integer	Number of 1xx responses
r2xx	Integer	“” “” “” 2xx responses
r3xx	Integer	“” “” “” 3xx responses
r4xx	Integer	“” “” “” 4xx responses
r5xx	Integer	“” “” “” 5xx responses
runk	Integer	“” “” “” responses with unknown HTTP status code
recorded_at	UNIX ts w/ frac	*Deprecated.* When this job was logged to ArchiveBot’s CouchDB
settings_age	Integer	Job settings version; incremented for each settings change
slug	String	WARC/JSON base filename [2]
started_at	UNIX ts w/ frac	When this job was started by a pipeline
started_by	String	The user (typically an IRC nick) that submitted the job
started_in	String	Where the job was started (typically an IRC channel)
suppress_ignore_reports	Boolean	Whether ignore pattern matches should be reported
ts	UNIX ts w/ frac	Last update received from a pipeline for this job
url	String	The URL for this job: either the target or a URL file (for !ao < and !a <)
user_agent	String	The user-agent to spoof; null if we should use the default agent

[1]: The expected relationship between these values is

last_analyzed_log_entry <= last_broadcasted_log_entry <= last_trimmed_log_entry

[2]: Usually looks like “twitter.com-inf”. The date, time, WARC sequence, extension, etc. are all appended by the pipeline.

2.6 IDENT_ignores

Type: set

Ignore patterns for the identified job. Each ignore pattern is a Python regex.

2.7 IDENT_log

Type: zset

Log entries generated for a job by the wpull hooks or pipeline stdout capture are sent here. The backend is notified of new entries in this set when the pipeline publishes the job ident on the `updates` channel.

2.8 pipelines

Type: list

Deprecated. This list contains pipeline names, and is still modified by pipelines, but no pipeline listing uses it.

2.9 jobs_completed, jobs_aborted, jobs_failed

Type: string

These keys store counts of completed, aborted, and failed jobs, respectively.

A completed job is a job that made it through the entire ArchiveBot pipeline. An aborted job is a job that was terminated using `!abort`. A failed job is a job that crashed and was reaped using the internal console.

2.10 tweets:done, tweets:queue

Type: zset

These are used by ArchiveBot's Twitter tweeter. They store tweets that were tweeted and tweets in the to-post queue, respectively.

2.11 Pubsub channels

2.12 updates

Whenever a pipeline has new log entries for a job, it publishes that job's ident to this channel.

2.13 archivebot:job:IDENT

There exists one of these channels per job.

When settings are updated for that job, the new settings age is published via this channel. The job's settings listener receives the new version. If the new version is greater than the current version, the new settings are read from Redis and applied.

CHAPTER 3

ArchiveBot Administration

ArchiveBot has a central "control node" server, currently run by Archive Team member David Yip (yipdw) at `archivebot.at.ninjawedding.org`. This document explains how to manage it, hopefully without breaking anything.

This control node server does many things. It runs the actual bot that sits in the EFnet IRC channel #archivebot and listens to Archive Team members' commands about which websites to archive. It runs the Redis server that keeps track of all the pipelines and their data. It runs the web-based ArchiveBot dashboard and pipeline dashboard. It runs the Twitter bot that sends information about what's being archived. It has access to log files and debug information.

It also handles many manual administrative tasks that need doing from time to time, such as cleaning out (or "reaping") information about old pipelines that have gone offline, or old web crawl jobs that were aborted or died or disappeared.

Another common administrative task on this server is manually adding new pipeline operators' SSH keys so that their pipelines can communicate with the dashboard and be assigned new tasks from the queue.

3.1 Basic Information

The control node server is reachable by SSH at `archivebot.at.ninjawedding.org`.

Archive Team members can SSH into this server with two possible usernames:

- `archivebot@archivebot.at.ninjawedding.org` - for performing more delicate administrative tasks
- `pipeline@archivebot.at.ninjawedding.org` - for adding/editing SSH keys for new pipeline servers

Neither of these accounts has sudo access.

Long-time Archive Team volunteers used to be assigned individual user accounts on this machine, but starting in mid-2017 all new pipelines are now added to the server via the shared `pipeline@` account instead, with a shared `authorized_keys` file, to keep things simpler.

This control node server is the same server that also runs the web-based ArchiveBot dashboard: http://dashboard.at.ninjawedding.org/

And it also runs the web-based ArchiveBot pipeline dashboard: http://dashboard.at.ninjawedding.org/pipelines

3.2 How to add new ArchiveBot pipelines

Archive Team volunteers set up and run pipelines on their own servers. Each of these can handle several web crawls at a time, depending on their servers' individual configuration and their available hard drive space and memory. More information and installation instructions are at GitHub: https://github.com/ArchiveTeam/ArchiveBot/blob/master/INSTALL.pipeline

When a new pipeline is set up and all ready to go, the last step is that the server's SSH key still needs to be manually added to the control node. The new pipeline's operator should e-mail or private message one of the Archive Team members who already has SSH access to the control node server, such as David Yip (yipdw), Brooke Schreier Ganz (Asparagirl) or Just Another Archivist (JAA), who may be hanging out in #archiveteam on EFnet. One of them should SSH into the `pipeline@archivebot.at.ninjawedding.org` account, and do:

```
`bash cd /home/pipeline/.ssh `
```

Then they should open the file `authorized_keys` with the text editor of their choice, and add the new pipeline server's SSH key to the bottom of the list, save, and quit. If the new pipeline is set up correctly, it should then show up on the web-based pipeline dashboard shortly after that, and should start being assigned web crawl jobs from the queue.

3.3 All about tmux

The control node server has many different processes running constantly. To help keep these processes running even when people log in or out, and to keep things somewhat well-organized, the server is set up with a program called `tmux` to run multiple "panes" of information.

When you log into the control node server, you should type `tmux attach` to view all the panes and easily move between them.

Here are some common tmux commands that can be helpful:

- Control-B N - moves to the next pane
- Control-B C - create a new pane
- Control-B W – select a pane/window (shows all running panes)
- Control-B [0-9] – go to a specific pane number (numbered 0 through 9)
- Control-B S – select an entirely different tmux session (although there should usually be just one)

Each pane has a process running in it, sometimes more than one process, for handling a different administrative task.

3.3.1 tmux pane 0: spiped (secure pipe daemon)

This pane runs `spiped` for Redis, which is used by some but not all pipelines. `spiped` is secure pipe daemon, and it forwards packets from one port to another port. The preferred connection is ssh tunneling.

Administrators probably won't need to do much in this pane, but it's useful to keep an eye on things.

3.3.2 tmux pane 1: pipeline manager

This pane runs the pipeline manager, which is `plumbing/updates-listener`. This listens for updates coming into Redis from all of the many pipelines. It then sends these updates to a ZeroMQ socket, which is what used by the web-based ArchiveBot dashboard (and possibly a few other things?); the dashboard is listening on publicly accessible port 31337.

(This port is *not* where the ArchiveBot Twitter bot gets its data; that's a different daemon.)

Logs from this pipeline manager are stored in `plumbing/log-firehose`. Someday this log firehose could be replaced with Redis pubsub.

3.3.3 tmux pane 2: pipeline log analyzer and log trimmer

This pane manages the pipeline log analyzer and the pipeline log trimmer.

The log analyzer looks at updates coming off the firehose and classifies them as HTTP 1xx, 2xx, etc, or network error.

The log trimmer is an artifact of how ArchiveBot stores logs, could probably be removed someday. It gets rid of old logs from Redis to prevent out-of-memory errors.

3.3.4 tmux pane 3: web-based dashboard

This pane runs the web-based ArchiveBot dashboard, which is publicly viewable at: http://dashboard.at.ninjawedding.org/

This tmux pane is split into two parts on the screen, top and bottom. The top pane shows the throughput of the dashboard web socket, which is the rate of data flowing from the log firehose to the dashboard.

The web-based dashboard has a small unknown memory leak, so the bottom pane runs and monitors ivan's "dashboard killer" daemon. It constantly polls the dashboard to see if it's alive, and it prints a dot if it was a success (dashboard was alive and responded). If the dashboard does not respond, probably because of that small memory leak, then this daemon kills it and automatically re-spawns it.

3.3.5 tmux pane 4: IRC bot

This pane runs the actual ArchiveBot, which is an IRC bot that sits in the channel #archivebot on EFnet and listens for Archive Team volunteers feeding it commands about what websites to archive.

Usually, there's not much that an administrator will need to do for this. If the bot gets kicked off EFnet, it will try to reconnect on its own. However, EFnet sometimes has the tendency to netsplit (disconnect from some IRC nodes in a disorganized manner). If that happens, the bot might try to rejoin a server that's been split, in which case the bot might need to be "kicked" (restarted and reconnected to the IRC server).

If you need to kick it, hit `^C` in this pane to kill the non-responding bot. Then hit the `Up arrow key` to show the last command that had been typed into bash, which is usually the one that invokes the bot. You can then adjust that command if you need to (such as possibly changing the server), and then hit enter to re-run that command and reconnect the bot to EFnet.

3.3.6 tmux pane 5: redis-cli console

This is the console for running redis-cli commands. It might get closed down, because it's rarely used.

3.3.7 tmux pane 6: job reaper and Twitter bot

This is the job reaper, used by administrators to manually get rid of "zombie" web crawl jobs that are dead or quit but which are still showing up for some reason on the web-based dashboard, cluttering it up.

Every job has a heartbeat associated with it, which Redis monitors. This pane will let you know if certain jobs' heartbeats have not been seen for a long time, which would indicate that the jobs are zombies.

If you need to reap a dead ArchiveBot job – in this case, one with the hypothetical job id 'abcdefghiabcdefghi' – here's what to do in this pane:

```
`bash cd ~/ArchiveBot/bot/ bundle exec ruby console.rb j = Job.
from_ident('abcdefghiabcdefghi', $redis) `
```

At this point, you should get a response message starting with `<struct Job...>`. That means the job id does exist somewhere in Redis, which is good. Then you should run:

```
`bash j.fail `
```

This will kill that one job, but note that the magic Redis word in the command here is 'fail', not 'kill'. This deletes the job state from Redis.

It is possible to reap multiple jobs at once, by mapping their job id's with regex and such. Such exercises are best left to experts.

You can also clean out "nil" jobs with redis-cli in the admin console with this command:

```
`bash idents.each { |id| $redis.del(id) } `
```

That command would send the delete command about each id to the Redis server.

This tmux pane 6 *also* runs the ArchiveBot Twitter bot connector. You shouldn't need to do anything with that most of the time, but it ever dies, go to pane 6 and press up and enter to re-run command, which is:

```
`bash bundle exec ruby start.rb -t twitter_archivebot.json `
```

The Twitter bot is publicly viewable at https://twitter.com/ArchiveBot/ .

3.3.8 tmux pane 7: couchdb

This pane inserts couchdb documents. You can probably ignore this, and should leave it as-is.

3.3.9 tmux pane 8: the pipeline reaper

This is the pane where you can reap old dead pipelines from the pipeline monitor. You can view the web-based pipeline monitor page here: http://dashboard.at.ninjawedding.org/pipelines

Pipeline data is stored inside Redis. You can get a list of all the pipelines Redis knows about with this command:

```
`bash ~/redis-2.8.6/src/redis-cli keys pipeline:* `
```

That will list all currently assigned pipeline keys – but some of those pipelines may be dead.

To peek at the data within any given pipeline – in this case, a pipeline that was assigned the id 4f618cfcd81f44583a93b8bdb50470a1 – use the command:

```
`bash ~/redis-2.8.6/src/redis-cli type pipeline:4f618cfcd81f44583a93b8bdb50470a1
`
```

To find out which pipelines are dead, check the web-based pipeline monitor and copy the unique key for a dead pipeline.

To reap the dead pipeline (two parts):

```
`bash ~/redis-2.8.6/src/redis-cli srem pipelines pipeline:4f618cfcd81f44583a93b8bdb504
`
```

That removes the dead pipeline from the set of active pipelines. Then do:

```
`bash ~/redis-2.8.6/src/redis-cli del pipeline:4f618cfcd81f44583a93b8bdb50470a1
`
```

NOTE: be very careful with this; make sure you do not have the word "pipelines" in this command!

That deletes that dead pipeline's data.

3.4 Re-sync the IRC !status command to actual Redis data

The ArchiveBot `!status` command that is available in the #archivebot IRC channel on EFnet is supposed to be an accurate counter of how many jobs are currently running, aborted, completed, or pending. But sometimes it gets un-synchronized from the actual Redis values, especially if a pipeline dies. Here's how to automatically sync the information again, from Redis to IRC:

```
`bash cd /ArchiveBot/bot bundle exec ruby console.rb in_working =
$redis.lrange('working', 0, -1); 1 in_working.each { |ident| $redis.
lrem('working', 0, ident) if Job.from_ident(ident, $redis).nil ? }
`
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search