Mantrid Documentation

Release 1.0

Epio Limited

March 08, 2013

CONTENTS

Mantrid is the HTTP load balancer used for Epio. It is designed with high availability and simplicity in mind: it is configured at runtime with JSON over HTTP and can temporarily hold open connections while backend servers restart. It monitors bandwidth and connection statistics and is ideal for serving large numbers of hostnames.

It trades some raw speed for flexibility, but is still designed to be fast. Its aim is to have latency of no more than 10ms, and have no more than a 10% reduction in throughput.

It is available on GitHub.

CHAPTER

ONE

INSTALLATION

If you haven't got pip installed, install it from a system package (`python-pip` on Ubuntu and Debian) or run:

```
$ sudo easy_install pip
```

Then run:

```
$ sudo pip install mantrid
```

You can improve performance by using PyPy 1.7 or greater. Just use the pypy-specific pip to install it. At the time of writing, PyPy 1.7 is not yet released, but a nightly build will work.

CHAPTER

TWO

QUICK START

To run Mantrid with a default configuration, just run:

```
$ sudo mantrid
```

(Or run `mantrid` as root.) Mantrid needs root in order to bind to port 80 and set its resource limits. It automatically drops to a less privileged user once it has started up.

The default configuration is to serve external clients on port 80 (from all available addresses), and to have management on port 8042 bound to localhost.

See the *Guide: A Simple Setup* article for a walkthrough of an initial, simple installation.

CHAPTER

THREE

CONFIGURATION

Mantrid will look for startup configuration in `/etc/mantrid/mantrid.conf` by default. You can specify an alternative location on the command line:

```
$ mantrid -c /home/andrew/mantrid.conf
```

The configuration file is in the format `variable_name = value` and comments are denoted by starting them with a `#`. For available configuration options, see the *The configuration file* page.

Note that the configuration file only tells Mantrid how to start up; configuring hostnames and Mantrid's responses are done via the REST API. You can use the included `mantrid-client` tool to interact with the REST API. For more information, read *Configuring rules*.

CHAPTER

FOUR

RUNNING AS A NORMAL USER

If you only make Mantrid listen on port 1024 or greater, there is no need to run it as root. Mantrid won't be able to automatically change resource limits as a normal user, but you can do it manually with things like `ulimit` or `pam_limits`.

CHAPTER

FIVE

TABLE OF CONTENTS

5.1 The configuration file

This file is looked for at either `/etc/mantrid/mantrid.conf` or the location passed on the command line using the `-c` switch.

The default settings would look like this:

```
bind = *:80
bind_management = *:8042
state_file = /var/lib/mantrid/state.json
uid = 4321
gid = 4321
static_dir = /etc/mantrid/static/
```

5.1.1 Address formats

Mantrid supports both IPv4 and IPv6, so bind addresses can be supplied in both formats:

```
bind = 10.0.0.45:80
bind = 0.0.0.0:80
bind = [fe32::54ab:12cc]:80
bind = [::]:80
bind = *:80
```

The special address `*` will bind to all connections on either IPv4 or IPv6.

5.1.2 Options

bind

Tells Mantrid to bind to the given address and port to serve external users. Use the address `*` to listen on all available addresses.

This option may be specified more than once to listen on multiple ports or addresses.

bind_internal

Tells Mantrid to bind to the given address and port to serve internal proxies. Use the address `*` to listen on all available addresses.

Requests from internal proxies will not have their `X-Forwarded-For` and `X-Forwarded-Protocol` headers removed. 'Internal' bind addresses are for use behind an SSL terminator, which should add these headers itself.

This option may be specified more than once to listen on multiple ports or addresses.

bind_management

Tells Mantrid to bind to the given address and port to serve management API requests. Use the address `*` to listen on all available addresses.

Note that there is no authentication on the Mantrid management API; anyone who can access the port can wipe your loadbalancer. We suggest you limit it to either local connections only or a secure subnet.

This option may be specified more than once to listen on multiple ports or addresses.

state_file

Specifies the location where Mantrid stores its state between restarts. Defaults to `/var/lib/mantrid/state.json`. Should be writable by the user Mantrid drops priviledges to; it will attempt to make that possible if it has root access when it is launched.

uid

The UID to drop to once Mantrid has started. Defaults to 4321.

gid

The GID to drop to once Mantrid has started. Defaults to 4321.

static_dir

The directory which Mantrid will look in for static response files (ending in `.http`) used by the `static` action. Defaults to `/etc/mantrid/static/`.

5.2 Configuring rules

All configuration of Mantrid's load-balancing rules is done at runtime using a REST API (either directly, or via the `mantrid-client` command line client). For simplicity, this document will just demonstrate using the command line client.

Mantrid only works on the basis of hostnames; for each incoming request, it will take its hostname and attempt to resolve it to an *action*. It will first try and find an exact match to the hostname, and if no match is found it will then keep removing the first part of the hostname (up to the first `.`) until it has consumed the entire hostname.

Partial matches will only occur if the domain that is eventually partially matched allows subdomain matches.

For example, if we asked for the host "www.andrew.example.com", Mantrid would try to find rules matching these hostnames (in order):

```
www.andrew.example.com
andrew.example.com
example.com
com
```

If there was an entry for `andrew.example.com` with subdomain matches allowed, this would match. If only exact matches were allowed, however, this would not match that entry.

Each rule is made up of three parts: an *action name*, arguments for that action (as keywords), and the "are subdomain matches allowed" flag. You can read about the *Actions* and see what options you have.

All changes made via the API take effect immediately, for all future requests.

5.2.1 Adding and updating rules

Adding and updating a rule is the same operation, called 'set'; if there's a previous record for the hostname you're setting, it will be overwritten. To add a rule that just returns an empty 403 Forbidden to everyone requesting "top-secret.com", or any subdomains, you would call:

```
$ mantrid-client set top-secret.com empty true code=403
```

The arguments are, in order, the host name (`top-secret.com`), the action name (`empty`), the subdomains_allowed flag (`true`), and the arguments (`code=403`, to tell the empty action what status code to use).

5.2.2 Deleting a rule

Deleting a rule is pretty simple:

```
$ mantrid-client delete top-secret.com
```

5.2.3 Listing rules

You can get a human-readable list of rules using:

```
$ mantrid-client list
```

This produces something like the following:

```
HOST                                ACTION                  SUBDOMS
top-secret.com                      empty<403>              True
www.forever.com                     spin                    True
```

5.3 Actions

Actions are what you build rules with in Mantrid - they cover all the tasks from returning error pages to proxying requests through to a backend (as you'd expect a load balancer to do).

Each action has zero or more configuration options - where no table of options is shown, the action in question takes no configuration options.

5.3.1 empty

Argument	Required	Description
code	Yes	The (numeric) HTTP code to send as a response

Sends a HTTP response with a zero-length body (literally just the status code and response headers).

5.3.2 proxy

Argument	Required	Description
backends	Yes	A list of backend servers to use
attempts	No	How many times a connection is attempted to the backends. Defaults to 1.
delay	No	Delay between attempts, in seconds. Defaults to 1.

Proxies the request through to a backend server. Will randomly choose a server from those provided as "backends"; provides no session stickiness.

If a connection to a backend drops, it can optionally retry several times with a delay until it gets a response. If no connection is ever accomplished, will send the `timeout` static page.

5.3.3 redirect

Argument	Required	Description
redirect_to	Yes	The URL to redirect to.

Sends a HTTP 302 Found response redirecting the user to a different URL. If the URL specified has the protocol part included, they will be forced onto that protocol; otherwise, they will be forwarded with the same protocol (http or https) that they are currently using.

Note that the use of HTTPS is detected by the presence of an `X-Forwarded-Proto` or `X-Forwarded-Protocol` header on a `bind_internal` interface. Mantrid cannot do SSL termination itself.

5.3.4 spin

Argument	Required	Description
timeout	No	How long to wait before giving up, in seconds. Defaults to 120.
check_interval	No	Delay between rule checks, in seconds. Defaults to 1.

Holds the incoming request open, checking Mantrid's rules table periodically for a match. If a new rule is added matching the hostname while the request is being held open, Mantrid will then pass control over to the new action. If no new rule is added before the timeout expires, sends the `timeout` static response.

This is particularly useful for webservers that are being started or restarted; you can set the site to `spin`, restart the webserver (knowing that your requests are being held), and then set the rule back to `proxy` again and all the requests will continue as normal.

5.3.5 static

Argument	Required	Description
type	Yes	The name of the static response to send, without the `.http` extension.

Sends a HTTP response that is already saved as a file on disk. Mantrid ships with several default responses, but you can provide your own in the directory specified by the `static_dir` configuration option.

Default responses:

- `no-hosts`, used by the `no_hosts` action (short message for a fresh mantrid install)
- `test`, a short test page that says "Congratulations!...".
- `timeout`, used by the `spin` and `proxy` actions after a timeout.
- `unknown`, used by the `unknown` action.

5.3.6 no_hosts

Sends a predefined response pointing out that the load balancer has no hosts configured at all. The default action if the server is completely devoid of rules (as it would be on a fresh install).

5.3.7 unknown

Sends a predefined response that says "The site you have tried to access is not known to this server". The default action for any unknown host; takes no arguments. It is unlikely you would want to set this as part of a rule.

5.4 REST API

Mantrid is configured mainly via a REST API, available on port 8042 by default. All changes are done using HTTP with JSON responses.

Note that a *rule* is always formatted as a triple of `[action_name, kwargs, match_subdomains]`, where `action_name` is a string, `kwargs` is a mapping of strings to strings or integers, and `match_subdomains` is a boolean.

Statistics are returned as a dictionary with four entries: `open_requests`, `completed_requests`, `bytes_sent`, and `bytes_received`. The names are reasonably self-explanatory, but note that the two byte measurements are only updated once a request is completed.

5.4.1 /hostname/

GET

Returns a dictionary with all hostnames and their rules.

PUT

Accepts a dictionary in the same format that GET produces (hostname: rule)

5.4.2 /hostname/www.somesite.com/

GET

Returns the rule for this hostname, or None if there is no rule for it currently.

PUT

Accepts a rule triple to be used for this hostname.

DELETE

Removes the rule for this hostname.

5.4.3 /stats/

GET

Returns a dictionary with all hostnames and their statistics.

5.4.4 /stats/www.somesite.com/

GET

Returns the statistics for just the specified hostname.

5.5 Guide: A Simple Setup

This guide will show you how to get a very simple Mantrid install working - we'll have one host, which we proxy through to a backend, and then we'll show you how to put it into "spin" mode (where it will hold open incoming connections) and then back into proxy mode, which will then let all the pending connections through.

First of all, install Mantrid; instructions on how to do this are on *the main page*. Once you have it installed, you need to start Mantrid:

```
sudo mantrid
```

Now Mantrid should be listening on port 80, and listening for management connections on localhost:8042. If you go to http://localhost/ now, you should get a simple page telling you that you currently have no hosts.

5.5.1 A single host

Let's add an example host - we'll just use "localhost" for now, and tell it to proxy to `google.com`:

```
mantrid-client set localhost proxy true backends=localhost:8000
```

That tells the client to set a new rule, for the domain `localhost`, using the action `proxy`, handling subdomains as well (`true`), and then specifies the one backend we're using - in this case, we're presuming you're running something on port 8000 locally - change that as required.

If you now go to http://localhost/, you should see the application you redirected to appear.

5.5.2 Holding back connections

Now, let's change localhost to 'spin' incoming connections:

```
mantrid-client set localhost spin true
```

(spin takes no arguments, so there is nothing after `true`).

If you now visit http://localhost/, your browser will just sit and try and load the page - Mantrid is holding open connections (this is useful if, for example, you are restarting your web servers). Now, you can set it back to proxy mode:

```
mantrid-client set localhost proxy true backends=localhost:8000
```

Your open connection will then successfully go through and serve the page you saw before.

5.5.3 Multiple backends

You can also set more than one backend; if we set:

```
mantrid-client set localhost proxy true backends=localhost:8000,localhost:8001
```

then hitting http://localhost/ will randomly connect you through to one of the two ports.