Snapt Balancer Manual

Version 2.1

17/08/2017

Contents

Chapter 1: Introduction 3
Chapter 2: General Usage 4
Configuration – Default Settings (Global Settings) 5
Configuration – Performance Tuning 12
Configuration – Lua Scripts 12
Configuration – Resolvers 12
Configuration – Snapt Settings 12
Configuration – Logging Options 13
Configuration – Connection Logging 14
Configuration - Peer Management 15
Configuration – Backups 15
Configuration – HTTP Error Management 15
Configuration – Initial Configuration 16
Configuration – Installation Wizard 17
Create a Load Balancer 18
Chapter 3: Group, Frontend and Backend Management 19
Group Management 20
Group Settings 21
Servers in Groups 31
Frontend Management 34
Backend Management 35
ACL Management 35
Chapter 4: Standard Operation 38
Chapter 5: Reporting 40

Chapter 1: Introduction

The Snapt Balancer is a feature-rich layer 7 TCP load balancer. This message conveys quite a lot of information which we will break down to begin with. If you have a solid understanding of load balancing you can progress to chapter 2.

There are two primary protocols on the internet – TCP and UDP. These are what we call layer 4 protocols and they don't by themselves suggest what they are actually doing – like web browsing, or email etc. The majority of the data sent across the internet is TCP and that is what Snapt load balances. Protocols like HTTP, SMTP, SSL and much more all use TCP.

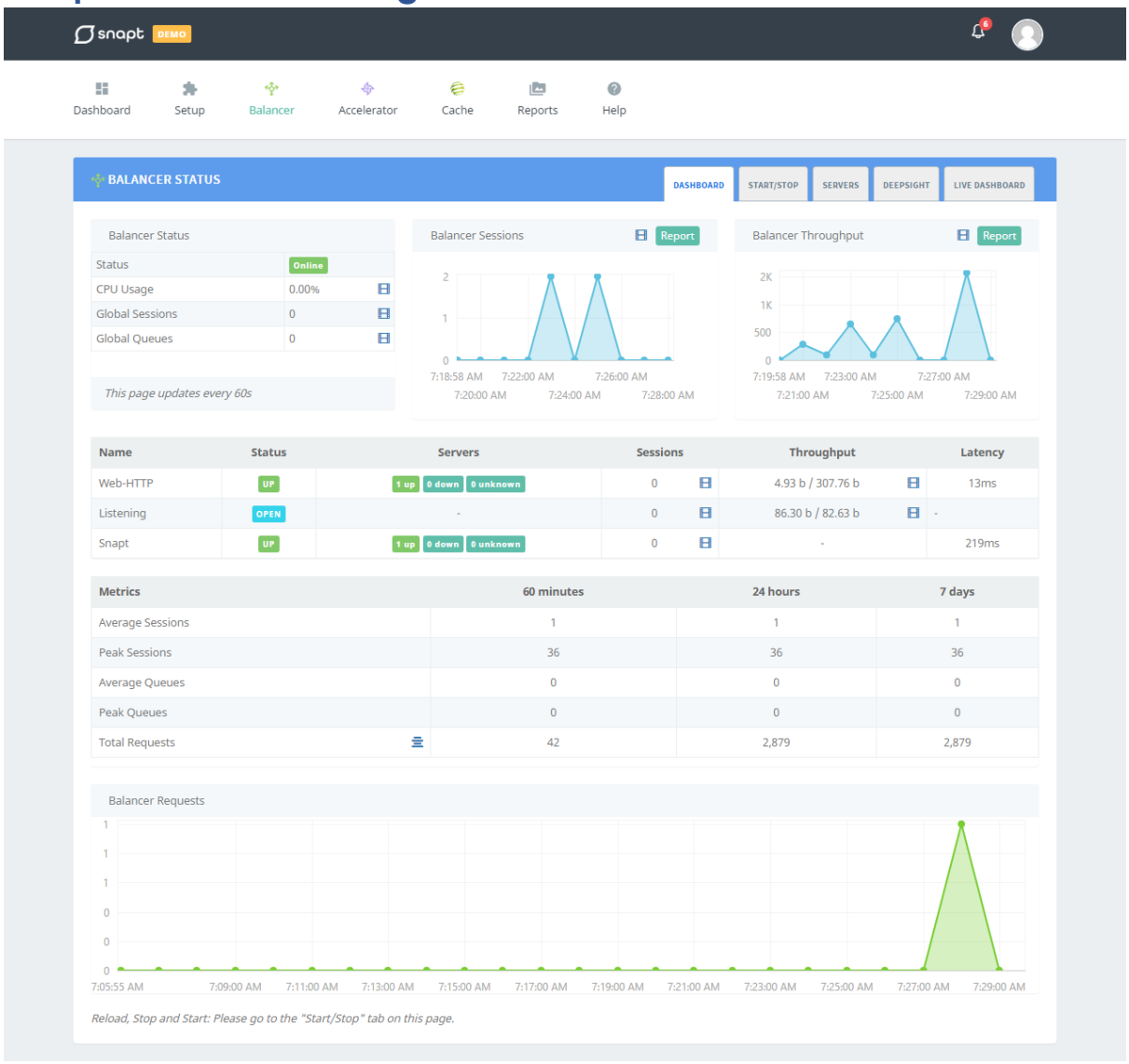
A load balancer is initially required when one of two things happens in your business – either your server becomes so busy that you need two (or more of them) or you need to ensure that even if your server fails another one will be online to take its place. In larger businesses, both are almost always true.

The load balancer sits in front of your servers and accepts connections from clients that would have normally been communicating directly with the servers, for example, web browsers. These web browsers instead connect to the load balancer and are then sent to one of the available servers. While doing this the load balancer is monitoring the servers to ensure they are online and working and balancing the traffic equally.

Layer 7 is where the smart stuff begins to happen. That's the networking layer at which protocols begin to mean something to the common person – for example, HTTP is a layer 7 protocol. All web browsing are communicating with either HTTP or SSL (HTTPS) to browse web content.

Load balancers like Snapt understand this, and instead of just looking at the TCP part of the request they analyse the HTTP part as well – allowing them to do clever things like send requests for images to certain servers, to mask error pages with maintenance pages, to create access control lists based on URLs and much more.

Chapter 2: General Usage

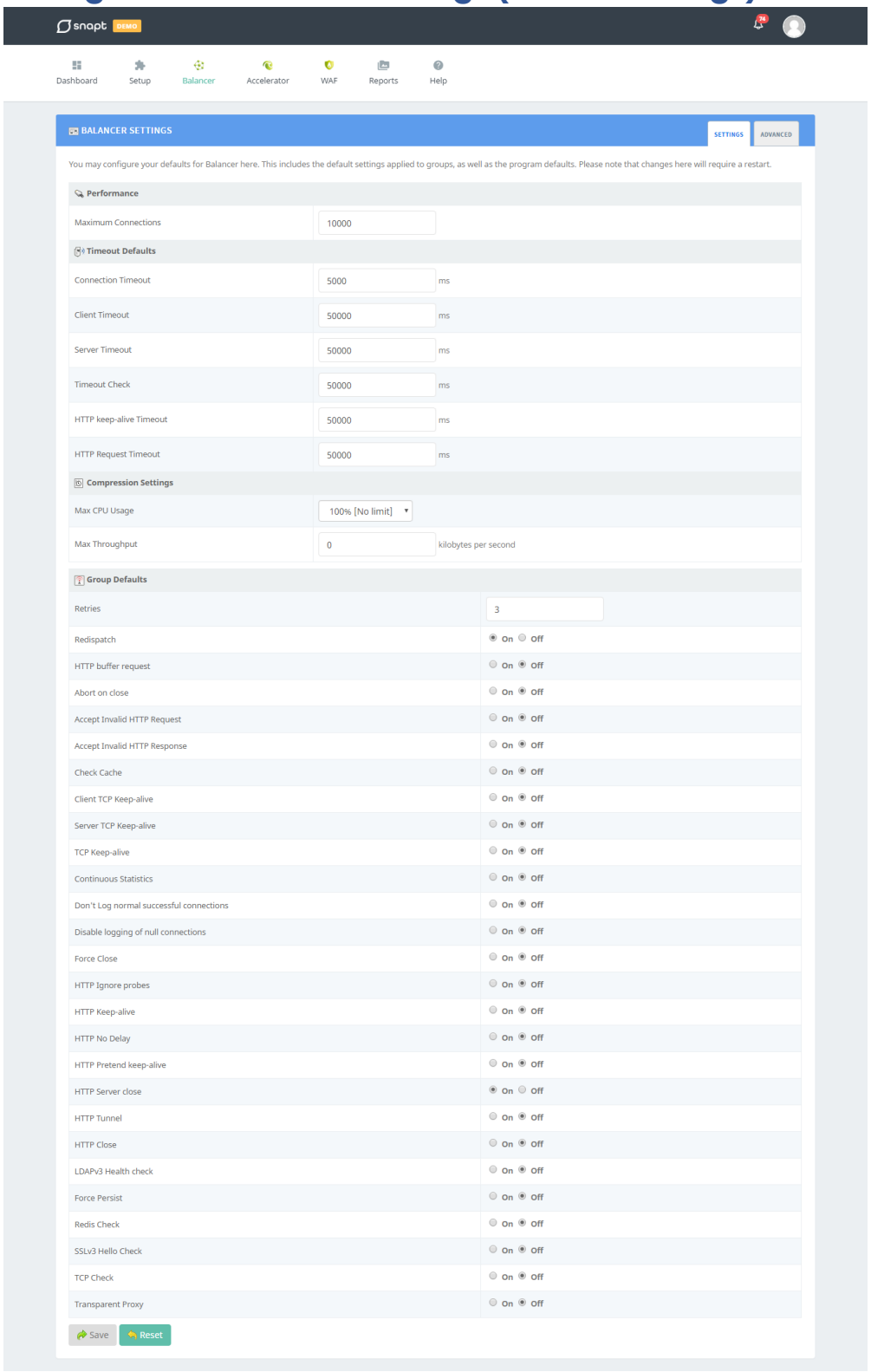


This manual assumes you have already setup your Snapt install and enabled the Balancer plugin. If you have not, please follow the Snapt Balancer – Getting Started guide.

Everything you will need to do regarding the Balancer happens under the Balancer menu item in the framework. The first main menu option is Configuration, and this holds all your global defaults for the load balancer.

In the Balancer, there are essentially 3 primary areas to familiarize yourself with. The global options as we are discussing now, the group or frontend and backend options which deal with actually load balancing a group of servers and the status sections which provide overviews and reporting information for your load balancer.

Configuration – Default Settings (Global Settings)



Your default settings control the limits and timeouts for this server. Limits are not a bad thing! They can prevent denial of service attacks and abusive programs like brute force password crackers. Try to remember to set reasonable limits for this reason.

Maximum Connections

This setting limits the maximum number of connections that will be accepted by the balancer (across all groups and servers) at any point in time. After these connections will be refused until space becomes available.

Connection Timeout

How long should the balancer allow before a connection attempt to the servers will stop. This is just the initial hello of the conversation and should be a reasonably short amount of time.

A default value of 5000ms (5 seconds) is recommended.

Client Timeout

How long should the balancer allow before a client's inactive connection is closed. This only takes effect if the client does not close the connection correctly.

A default value of 50000ms (50 seconds) is recommended.

Server Timeout

How long should the balancer allow before a server's inactive connection is closed. This only takes effect if the server does not close the connection correctly.

A default value of 50000ms (50 seconds) is recommended.

Timeout Check

How long should the balancer allow before another timeout check is done after a connection is established.

HTTP Keep-alive Timeout

The purpose of this setting is to set a threshold on the number of queued connections at which the balancer stops trying to reuse the same server and prefers to find another one.

HTTP Request Timeout

The purpose of this setting is to set the maximum allow time to wait for a complete HTTP request.

Compression Max CPU Usage

You are able to disable compression when a certain CPU load level is reached. This allows you to stop compression at X% and re-enable it when the CPU goes below that. The purpose of this is to handle bursts of traffic and periods of high load as compression is a CPU intensive task.

The default is 100%, or disabled.

Max Throughput

Very similar to the Max CPU Usage setting the Max Throughput is the speed at which we will disable compression (optionally). Again this is to reduce the load on the server at times of high traffic.

The default is 0, or disabled.

Retries

Retries is the default value for how many times a connection to a live server will be retried before moving it to another server. This occurs in the event that a user is using a server which then fails. Already dead servers are never used.

A recommended value of 3 is the default.

Redispatch

Allow or disallow a session to be redistributed to another server when a server fails mid communication. This will occur after *Retries* has ended.

We recommend enabling this option.

HTTP buffer request

The balancer can ask the kernel not to immediately send an empty ACK upon a connection request, but to directly send the buffer request instead. This saves one packet on the network and thus boosts performance. It can also be useful for some servers, because they immediately get the request along with the incoming connection.

Abort on close

Allow or disallow a request to be served even if the request was aborted by the user.

Accept Invalid HTTP Request

Allow or disallow an invalid request to be processed. This option should never be enabled by default as it hides application bugs and open security breaches. It should only be deployed after a problem has been confirmed.

Accept Invalid HTTP Response

Allow or disallow an invalid response to be processed. This option should never be enabled by default as it hides application bugs and open security breaches. It should only be deployed after a problem has been confirmed.

Check Cache

Enables or disabled deep inspection of all server responses for strict compliance with HTTP specification in terms of cacheability. If a response doesn't respect these requirements, then it will be blocked. Due to the high impact on the application, the application should be tested in depth with the option enabled before going to production. By default this option has been disabled.

Client TCP Keep-alive

Enabled or disable the sending of TCP Keepalive packets on the client side. It is important to understand that keep-alive packets are neither emitted nor received at the application level. It is only the network stacks which sees them.

Server TCP Keep-alive

Enable or disable the sending of TCP Keepalive packets on the server side. Enabling socket-level TCP keep-alives makes the system regularly send packets to the other end of the connection, leaving it active. The delay between keep-alive probes is controlled by the system and depend both on the operating system and its tuning parameters.

TCP Keep-alive

Enable or disable the sending of TCP keepalive packed on both sides. Enabling socket-level TCP keep-alives makes the system regularly send packets to the other end of the connection, leaving it active. The delay between keep-alive probes is controlled by the system and depend both on the operating system and its tuning parameters.

Continuous Statistics

By default, counters used for statistics calculated are incremented only when a session finishes. It works quite well when serving small objects, but with big ones or with A/V streaming, a graph generated from HAProxy counter looks like a hedgehog. With this option enabled counters get incremented continuously

throughout the session. This option is not enabled by default, as it has a small performance impact (~0.5%).

Don't Log normal successful connections

When dealing with large sites with several thousand connections per second logging can be a major pain. Enabling this ensures that normal connections, those which encounter no error, no timeout, no retry nor redispatch, will not be logged. This leaves disk space for anomalies. In HTTP mode, the response status code is checked and return codes 5xx will still be logged. By default this option has been disabled in the balancer.

Disable logging of null connections

Allow or disallow null connections to be logged. In certain environments, there are component which will regularly connect to various system to ensure that they are still alive. It can be the case from another load balancer as well as from monitoring systems. By default, even a simple port probe or scan will produce a log. This can cause the logs to become polluted.

Force Close

Some HTTP servers do not necessarily close the connections when they receive the "Connection: close", and if the client does not close either, then the connection remains open till the timeout expires. This causes a high number of simultaneous connections on the servers. When force close is enabled is will actively close the outgoing server channel as soon as the server has finished responding and release some resources earlier.

HTTP ignore probes

Recently some browsers started to implement a "pre-connect" feature which results in speculatively connecting to some recently visited websites just in case the user would like to visit them. This results in many connections being established to web sites, which end up in 408 Request Timeout if the timeout strikes first, or 400 Bad Request when the browser decides to close them first. These ones pollute the log and feed the error counters.

HTTP Keep-alive

Allow or disallow HTTP keep-alive mode on the client and server-side. This provides the lowest latency on the client-side and the fastest session reuse on the server-side at the expense of maintaining idle connections to the server.

HTTP Server close

Allow or disallow HTTP server close mode. Server-facing connections are closed after the end of the response is received, but the client-facing connection remains open

HTTP Tunnel

Allow or disallow any HTTP processing past the first request and the first response. It is the mode with the lowest processing overhead, which is normally not needed anymore unless in very specific cases such as when using an in-house protocol that looks like HTTP but is not compatible, or just to log one request per client in order to reduce log size.

HTTP close

If this option is enabled the balancer will work in HTTP tunnel mode and check if a "Connection: close: header is already set in each direction, and will add one if missing. Each end should react to this by actively closing the CP connection after each transfer, thus resulting in a switch to the HTTP close mode. Any "Connection" header different from "close" will also be removed.

LDAPv3 Health check

When this is enabled, an LDAPv3 anonymous simple bind message is sent to the server, and the response is analysed to find an LDAPv3 bind response messages.

Force Persist

If this option is enabled, requests are not dispatched to down servers. It is possible to force this, but it is unconditional and redispatches to a valid server if Redispatch is enabled.

Redis Check

It is possible to test that the server correctly supports REDIS protocol instead of just testing that it accepts the TCP connection. When this option is set, a PING redis command is sent to the server, and the response is analysed to find the "+PONG" response message.

SSLv3 Hello Check

If this option is enabled, pure SSLv3 client hello messages are sent once the connection is established to the server, and the response is analysed to find an SSL server hello message. The server is considered valid only when the response contains this server hello message.

TCP Check

If this option is enabled a health check method is intended to be combined with "tcp-check" command lists to support send/expect types of health check sequences.

Transparent Proxy

If this option is enabled the balancer will provide layer 7 persistence to layer 3. The idea is to use the OS's ability to redirect an incoming connection for a remote address to a local process (HAProxy), and let the process know what address was initially requested.

SSL Server verify

If enabled all servers are tested till there correctly reply to SSLv3 client hello messages, and most servers tested do not even log the requests containing hello messages, which is appreciable.

Configuration – Performance Tuning

The performance tuning section is to do with operating system performance values and denial of service tweaks that can be made. These settings modify the actual Linux kernels values to suggested Snapt values in order to increase the performance available from a system.

Each option has a tooltip available and is a default setting by Snapt. You are able to manually control these "sysctl" values if you are an advanced user. If not, you can enable the various options to use the Snapt recommended defaults.

Configuration – Lua Scripts

In this section, you can manage, create and edit Lua Scripts. Lau Scripts can be used to execute custom actions or codes according to your needs. Please see https://www.lua.org/ for more information regarding Lua Scripts

Configuration – Resolvers

Resolvers can be attached to multiple server's definitions on group's and backends' and are reusable. This is not requested in default configuration.

Configuration – Snapt Settings

Snapt settings contain options for the user interface and specific Snapt tools which can be enabled or disabled. These can have performance effects on your system.

Auto-Reload Balancer

When enabled this will allow certain portions of the interface to automatically reload the Balancer to apply changes you have made. This speeds up use of the device but disables the ability to make all of your changes before doing a reload.

This is disabled by default.

Live Dashboard Interval

Enabling this will turn on live graph support on the Balancer Dashboard. While this can be great to have, it is often not used and can slow down the browsing experience when using the UI.

These are disabled by default.

Snapt Insights

Snapt Insights automatically tracks the usage and activity of every single IP address connecting to your server and displays various things using that information. Primarily this includes a user map on the Dashboard and a new submenu called Insights where you can browse the data.

This is disabled by default.

Configuration – Logging Options

Logging is a core component of the load balancer and is split into two parts. The standard logging options menu controls your performance logging and reports for the servers and the notifications of events.

Enable Logging

Turning this on will enable the automatic logging and reporting of information like sessions, queues, throughput, errors and more for all of your servers, groups, frontends and backends.

We advise enabling this, but it is disabled by default.

Log Frequency

Each x minutes we will store a log line for each server you have in the load balancer. The reason you are able to specify how often this should be done is because your database can grow very quickly. For example, if you have 20 servers in 5 groups and you log every minute you will generate 1.3 million logs per month.

For larger numbers of groups and servers, we advise 15 or 30-minute logging intervals, for smaller sites 5 minutes is advised.

Notice Events

Events are status changes in a server – for example, the server going DOWN or coming UP, or the group or frontend changing status. These are considered critical alerts as they may indicate a fault in your network or at least on a server.

We advise enabling this, but it is disabled by default.

Notice Level

Often servers can flap – this means to go down and up – and notifications can be a hassle. You are able to change the Notice Level in order to limit what type of notifications you will receive for your servers and groups.

We advise starting with all notifications on and fine-tuning after launch.

Configuration – Connection Logging

Connection logging records a log line for every request that goes through the load balancer. Because of the enormous amount of logs this creates we do not provide true on-box storage of them but rather allow you to use syslog – a network-capable logging protocol – to receive the logs.

Enable Connection Logging

This is disabled by default, and enabling it will begin logging every request with various details such as IP, URL, load times, date, etc.

Syslog Host

This is the syslog server and port that we should send the logs to. If you choose to use localhost the maintenance and setup of the syslog itself are not handled by Snapt.

HTTP or TCP Logging

There are two log formats available – TCP and HTTP. If you have non-HTTP traffic you must use TCP mode which gives less information than is available with HTTP traffic.

Syslog Facility

Which facility should we use when sending syslog messages. This is a part of the syslog protocol and requires an understanding of syslog.

We advise using the default, local0.

Log All

This option is enabled by default. Disabling this will only log on groups, and frontends that you set logging on.

DeepSight

DeepSight is a new intuitive way of monitoring your web servers for HTTP Reply Time, Server Connect Time, Queue Time and Client Request Time. This will give you a visual representation of the health of your servers.

Configuration - Peer Management

Peer Management allows you to create groups that can be used to synchronize stick tables between multiple Balancer nodes. This is useful in a redundancy situation or a multi-master setup.

Configuration – Backups

Snapt backups allow you to see and keep record of your last 20 changes that you've made. In this section you can quicky restore to previous configurations should you need to. You can also delete and preview your backups. Another useful aspect of backups is the ability to export your HAProxy config file should you need to.

Configuration – HTTP Error Management

HTTP Error Management allows you to insert a fake maintenance page in the event that an HTTP error is going to reach a client. This is a masking technique to prevent users from seeing an error.

For example, if all of your webservers go down then the load balancer has no choice but to display an error. Possibly some bad code has been rolled out to the servers and is making an error on every server. These are situations where we can replace the error with a maintenance message asking the user to try again later and alerting you.

This is disabled by default.

Configuration – Initial Configuration

When you first install and navigate to the Balancer plugin you will be redirected to the Initial Configuration page(unless you are using a Snapt VM image). Here you will be able to integrate Snapt with an existing HAProxy installation or be instructed to install the HAProxy package.

Configuration – Installation Wizard

Here you will be able to modify the default settings for the balancer, some Snapt general settings and Logging options.

The Wizard will take you through multiple sections for configuring Snapt. You may choose to change any of the values to suit your specific requirements, but you may leave them as defaults and continue onto the next section.

Configuration – Balancer SSL

SSL Options

In this section, you can specify the maximum number of SSL connections the Balancer will accept. You can also decide if you want to enable or disable Stronger DH Exchange.

By default Stronger DH Exchange is disabled.

Views & Data

Metric Monitor

The metric monitor provides you with a super powerful overview of the Snapt system itself, as well as individual web servers running behind your Snapt install.

All this information is graphed.

Watchpost

The Watchpost keeps an eye on all of your servers. It will prompt you immediately if any of your servers go down or any errors occur.

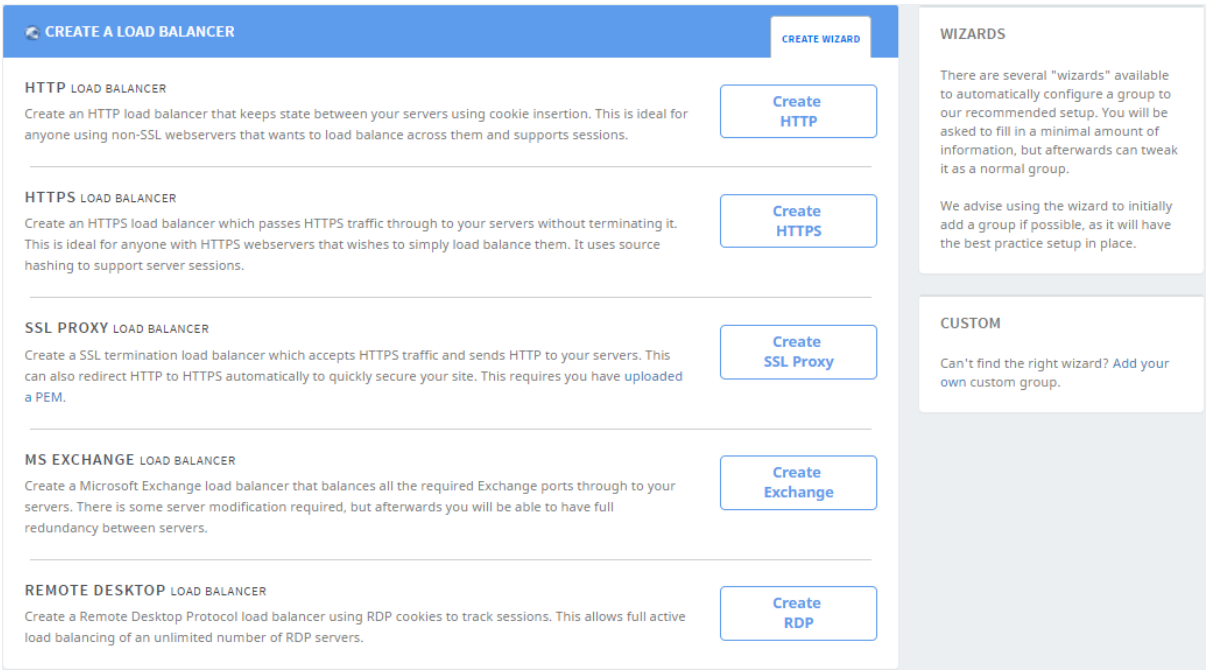
The Watchpost will alert you with a notification sound and will also display a red warning to make sure you're alerted to any issues.

Traffic Watch

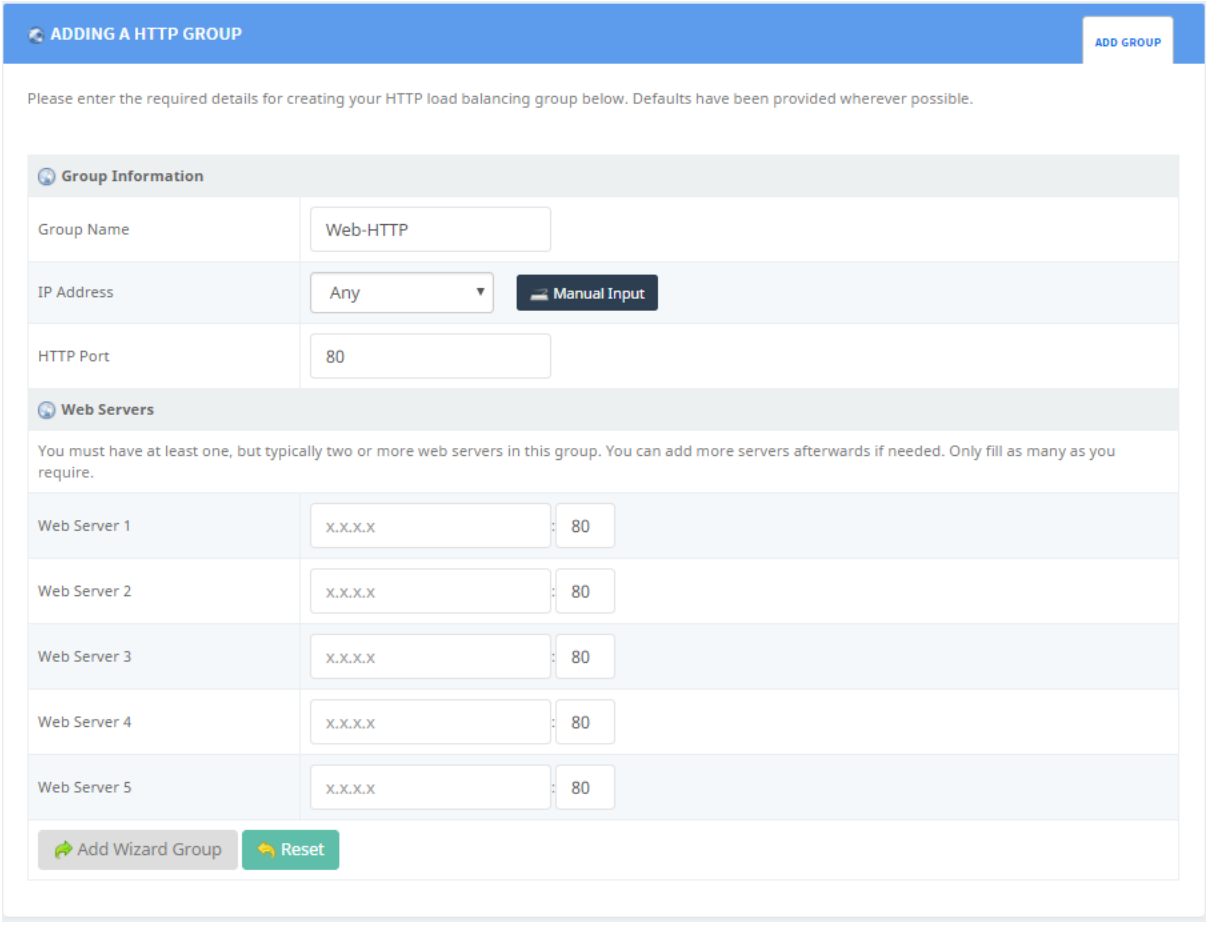
Traffic Watch provides you with an overview of your traffic statistics, compression savings and overall server health.

- **Traffic:** Current and Peak sessions, request rate and data volume in/out per server.
- **Compression:** Data savings due to compression.
- **HTTP:** A tally of status codes for requests per server.
- **Health:** The current status of each server based on health checks, warnings and errors.

Create a Load Balancer



This Wizard will guide you through creating a load balancing group. We provide templates for five of the most commonly used group types but you can add your own custom group if none of these suit your requirements.



Tooltips are available when you hover over each item in the wizard.

Chapter 3: Group, Frontend and Backend Management

The core of the load balancer is the groups. A group is made up of two parts – the listening port and IP and the destination servers.

When you create a group you will tell it to accept data on a port, for example 80 will be for HTTP traffic. You can then choose its options and proceed to add servers that the group should dispatch incoming requests to.

A frontend and a backend are these two parts, separated. The frontend accepts connections and the backend dispatches them. This is useful when you have one frontend but multiple backends that traffic must go to based on rules. For example, PHP content may go to one backend and images to another.

Group Management

The group management menu option provides you with a list of your current groups for editing and adding servers as well as a tab to add new groups. The options on this page allow you to *Edit* groups setting, *Disable* the group, *Delete* the group or manage the *Servers* inside the group.

When adding a group, the following options are available –

Group Name

The group name is a display name for your management and reporting needs and does not affect the actual traffic or service at all.

IP Address

This is the IP address the group will listen on. Often in redundant situations, you must choose the floating address, but for the simpler setups, you can select Any to listen on all IP addresses on the system.

The default is 0.0.0.0 or Any, which will listen on all available IP addresses including redundant ones.

Port Number

Which port this group should listen on. This is critical as it is where clients will send data to in order to load balance. For HTTP this would be 80, for HTTPS 443, and so on.

Group Settings

Whether you are adding a group or editing a group the options displayed will be the same. These options are also all available in frontends and backends but split between them.

Standard Options

Balance Options

Name

This is the name of the group you specified during setup by making use of the wizard or by manually adding the group.

Balance Method

The balance method selects the algorithm type to use when balancing incoming connections. Typically, this will be either Round-Robin or Least Connections. However, in situations where you require sticky sessions and cannot use cookie insertion you can select Source Hash.

- Round-robin: this will dispatch the connections in a round-robin fashion as they arrive. It is the smoothest and fairest method with short-connections like HTTP.
- Least Connections: this will send traffic to the server with the least active connections. This is only effective with protocols that have long communication times like LDAP, SQL and the like.
- STATIC RR: just like Round-robin, each server is used in turn, but with STATIC RR you can change the weight ratio for a server on the fly and will have no effect. However, you can define how many servers you like with this algorithm. In addition, when a server comes online, this algorithm ensures that the server is immediately reintroduced into the farm.
- First Server: this algorithm will first send all traffic to the lowest numeric identifier and will only when the first server's max connections are reached move to the next available server with the lowest numeric identifier. Should connections on the first server open up, new connection will then be processed on that server.
- Source Hash: this will hash all the available IPs to split them between the servers in a fixed way. The same IP address will always be sent to the same server.
- RDP Cookie (Advanced): this is specifically designed for the RDP protocol and will maintain session stickiness for RDP users.
- Host Header (Advanced): similar to Source Hash but with a hash on the host header this mode ensures that a website hostname always goes to the same server. This is useful when you manage a large amount of web domains and don't want to use additional resources.

- Custom (Advanced) : Custom allows you to manually specify some of the more advanced HAProxy balance options. *This is an advanced feature and not advised by Snapt*.
- Disabled (Unset): Certain custom configurations require no balance method to be set. This is for advanced users only

The default is round-robin, and is recommended for most situations.

Balance Mode

The balance mode instructs the load balancer how to balance this group and is relatively simple. For HTTP traffic, you will select HTTP and TCP traffic you will select TCP. Health mode is used to reply to external components health checks. This mode is deprecated and should not be used anymore as it is possible to do the same and even better by combining TCP or HTTP modes with the "monitor" keyword.

It is always better to select the more in-depth mode as it adds additional features, but cannot function on non-HTTP traffic.

Max Connections

This limits the number of connections the group will process at any one time. This limit will stop new clients being able to access your servers when hit, but should still be considered carefully as it prevents Denial of Service attacks from overloading your servers and limits abuse.

Source IP

This allows you set bind to a source IP that exists on the server when sending data to your backend servers. This is an advanced feature and not typically used.

This is disabled by default, meaning it will use the primary IP of the server.

Sessions Rate

You can limit the number of sessions per second using this option. Primarily focussed on preventing brute force attacks and denial of service attempts, this feature has the ability to limit legitimate clients and is for advanced users.

This is disabled by default.

Bind Addresses

Address List

This is an optional list of additional addresses the load balancer should listen on. For example, you may want to accept traffic on port 80 (your main port) but also port 81. In that situation you could add "0.0.0.0:81" to the list.

Access Control

TCP request content

Perform an action on a new session depending on a layer 4-7 condition. A request content can be analysed at an early stage of request processing called TCP content inspection. During this stage, ACL-based rules are evaluated every time the request contents are updated.

Backup Servers

All Backup

Enabling this will mean that all of your backup servers come online in the event of a server failure. The default (off) behaviour is to only bring up as many as have gone down.

This is disabled by default.

Group Timeouts

Custom Timeouts

You can override the timeouts you configured in the defaults section specifically for this group if required.

Stick Tables

Type

IP: This option only allows you to store IPv4 addresses.

Integer: This option allows you to store a substring of up to ten characters.

IPv6: This option allows you to store IPv6 addresses.

Size

Specifies the stick table size.

Expire

Defines the maximum duration an entry may exist in the table

Store

Allow you to set which data types may be stored.

Peer Group

If you configured Peer Group under Configuration > Peer Management you can select them here.

Protocol Tests

Protocol tests govern how the load balancer tests the servers to determine if they are online or offline. Only online servers are given data.

SSL Check [SSLv3 Only]

Enabling this will send an SSL v3 hello to the server as part of the health check. This is for testing SSL (HTTPS) servers.

LDAP Check

Enabling this will send an LDAP hello to the server as part of the health check. This is for testing LDAP servers.

Redis Check

Enabling this will test that the server correctly talks REDIS protocol instead of just testing that it accepts the TCP connection.

HTTP Check

Enable a more advanced HTTP (usually port 80) health check which checks that a valid HTTP reply is received from the server. You can additionally instruct it to consider the server down if a 404 (page not found) is received.

PGSQL Check

Enabling this option will send a PostgreSQL StartupMessage and waits for either Authentication request or Error response message.

HTTP Expect

By default this option considers that response status 2xx and 3xx are valid, and that other are invalid. When this option is used, it defines what is considered valid or invalid. Only one check statement is supported. If a server fails to responsd or times out, the check fails.

SMTP Check

Check for a valid SMTP server reply from the server. This is for SMTP servers typically running on port 25.

MySQL Check

Send an advanced MySQL check to the server, which attempts a login.

HTTP Options – Section

HTTP Options

HTTP options are only usable on an HTTP group (Balance Mode). These allow you to do HTTP specific actions to data which can enhance your load balancing for the protocol.

Force HTTP Close

We advise this is always enabled in order to allow fair load balancing. The option will insert a "Connection: Close" header on all HTTP requests going through to load balancer to ensure that each request only contains one object. Without this a client can send many requests in one session all of which go to the same server.

This is disabled by default, but should be used in almost all HTTP situations.

Add X-Forwarded-For

The X-Forwarded-For header is used to pass information on the original IP address which requested the page. Your webserver will see the Snapt Load Balancer IP instead of the client IP – enabling this option causes Snapt to add a header with the client IP for your servers use.

This is disabled by default, but is advised for HTTP use.

Compression Options

Gzip Compress

Enabling GZip compression will compress all the HTTP data which flows through the load balancer in order to reduce the size of pages going to your clients.

This is disabled by default.

Quick Change

Quickly change from compressing just pages to compressing all the Snapt Recommended.

Compression Content Type

A list of content types which the load balancer should compress. Unless you are an advanced user it is advised that you use the "Snapt Recommended" types.

The reason for this specification is because it is a waste to compress already compressed objects like ZIP files.

Offload Mode

Many web servers have compression built into them. However, compression is a CPU intensive task and offload mode allows you to force the webserver not to compress so that the load balancer can take the task on its behalf.

This is disabled by default, but typically it can be more efficient on the load balancer than your webservers. If you are using the accelerator do not enable this, and rather use the accelerator gzip.

HTTP Redistribution

Redispatch

Allow or disallow a session to be redistributed to another server when a server fails mid communication. This will occur after *Retries* has ended.

Persist

In the event where a cookie is instructing the load balancer that a user should go to a DOWN server should we honour that initially, or immediately redispatch the user.

Header Modification

HTTP buffer request

It is sometimes desirable to wait for the body of an HTTP request before taking a decision. Enabling this option will wait for the whole HTTP request body before processing.

Request Add

This option allows you to insert a header into all requests that are made to your servers, and in an ACL situation if required.

For example, you may add "X-Snapt-LB: Yes" as an instruction to your webservers.

Request Delete

You can delete a header from incoming requests here, and based on an ACL if required. This will prevent clients from sending the specified header to your webservers.

Redirects

Allows you to perform HTTP redirects based on certain parameters.

HTTP request rules

Defines a set of rules which apply to layer 7 processing. The rules are evaluated on their declaration order when they are met in a frontend, listen or backend section. Any rule may optionally be followed by an ACL-based condition.

Request Replace Rules

Defines a set of rules which apply to layer 7 processing. The rules are evaluated on their declaration order when they are met in a frontend, listen or backend section. Any rule may optionally be follow by an ACL-based condition.

Capture Request header

Allows you to capture request header to be stored in variable for later use. Data stored can be accessed using &[capture.reg.hdr(x)] variable in the log format

Capture Response header

Just like Capture Request Header. Capture Response Header allow you to store the response header for later use.

HTTP Only Options

Monitor URI

A URL which will be intercepted to return the status of the group, for example, /monitor.

Cookies

Cookies are a system used to store information on the clients web browser so that we may track which initial web server they went to.

For example, if you have a service where users log in to the website and we then send them to another server they may be logged out on that server. The cookie will store the server they first went to so that we may send them back to that server.

This is disabled by default.

Insert Cookie

This will enable the cookie insertion behaviour.

Cookie Name

A cookie name for our custom cookie. This can either be unique, for example "SNPLBID" or a cookie that you know exists on your website which we can hijack for our information.

Cookie Domain

Should you have a multi-domain service where users go to sites like client.snapt-ui.com and www.snapt-ui.com you can specify .snapt-ui.com as the cookie domain. This will share the information across both sites.

Cookie Options

Cookies options instruct the balancer on how it should use cookies. These are advanced settings. For standard users you should always tick "Insert" and nothing else.

Custom Configuration

Advanced Option

Allows you to add advanced options like abortonclose for example.

Advanced Override

Allow you to add advanced override options like http-server-close for example.

SSL options

We advise that you make use of the wizard, or manually creating a frontend/backend pair for SSL termination. Using a group for it is not advised.

SSL Termination

Allows you to terminate SSL sessions on the balancer rather than on your web servers. This will free up resources on your web servers and allow you to inspect SSL traffic before dispatching to your web servers. By default this is disabled.

Disable SSLv3

Allow or disallow SSLv3. By default this is disabled.

Bind

Here you can choose which IP and port to bind to for SSL.

Disable TLS 1.0

Allow or disallow TLS 1.0 connections. This is disabled by default

Disable TLS 1.1

Allow or disallow TLS 1.1 connections. This is disabled by default.

SSL Certificate

Here you can select which SSL certificate to use.

Client Certificate

If you have Client certificates loaded onto your Snapt system. You can then select them here.

CA Certificate

If you have loaded a CA certificate onto your Snapt system. You can select it here.

Allow SNI

Allow or disallow SNI. This allows a server to present multiple certificates on the same IP address and TCP port number and hence allows multiple secure websites to be served off the same IP address without requiring all those sites to use the same certificate.

Servers in Groups

Every group requires servers in order to know where to send incoming requests. These are your actual servers that you wish to load balance, for example, your webservers running on port 80.

They will be monitored for your configured health checks and if they are considered healthy, data will be dispatched to them.

On the servers page you can *Delete* or *Pause* existing servers and *Edit* each server's settings.

Add a Server

The add a server tab allows you to add a new server to a group. The IP and port details are also what the load balancer will use to do health checks.

Server Name

This is a display name for the server in reports and alerts and does not affect the actual operation of the server.

IP Address

What is the IP address of your server that we are adding.

Port

Which port is your server listening on?

Server Options

Each server can have unique options set like unique cookie strings or specific performance metrics.

Standard Options

Server Type

Is this a Primary server (the default) or a Backup server. Backup servers are only used when the primary servers are down.

Cookie String

A custom string (e.g. www01) is used to identify this server when using HTTP Cookies on your group.

Re-encrypt [SSL]

If you are terminating SSL requests on this load balancer you may want to re-encrypt SSL traffic to the webserver. In this setup ensure you are dealing with port 443 type traffic.

HTTP Redirect

You may issue an HTTP redirect to any clients that arrive at this server here. For example, you can tell clients to go to https://www.yoursite.com or if it is a backup server perhaps your DR location.

Health Checks

You may customize the frequency and rise and fall values of a server here

Force SSL Checks

When you've got an HTTPS/SSL setup, you may want to use SSL protocol for server health checking. Some server setups will not respond to health checks on port 80

Fall Count

How many bad responses to health checks should we allow before considering the server as DOWN.

The recommended default is 3.

Rise Count

How many good responses to health checks will we require before marking the server as UP again.

The recommended default is 5.

Interval

How often in milliseconds should we poll this server to verify its health.

The recommended default is 2000ms (every 2 seconds).

Health Check Port

Should you wish to do health checking on another port, you can specify the port here.

Resolvers

If you wish to use Resolvers, you may click the link "Click here to create a resolver" to setup one.

Performance

If you have servers with different performance specifications you may want to set different weights or maximum connection counts.

You may set those here.

Max Connections

A per server connection limit which is disabled when set to 0. This allows you to cap the number of active connections a server will have. If all the servers in your group reach this value additional clients will be rejected.

This is disabled by default.

Max Queue

The maximum amount of queued requests for this server, also disabled by setting it to 0.

This is disabled by default.

Weight

A server weight to be assigned to this specific server. This will cause allocations within the group to favour servers with higher weights. The higher the number the larger the percentage of traffic that server will receive. This allows you to have different sizes of servers participate fairly in the group.

The default is 10.

Source Binding

In the event, you want to bind to a specific set of source ports on a specific IP you can enter that here in the form x.x.x.x:1025-65000. This is useful when you want to avoid TCP exhaustion and use a unique IP per server.

Frontend Management

A frontend is the first half of what you know as a standard group – the frontend listens on an IP for incoming connections. This frontend then needs to make a decision on which backend to send that data to.

The options within the Frontend are identical to the Group Management options at the start of chapter 3, bar the Default Backend option.

Default Backend

This is where traffic not controlled by an ACL will arrive. It is wise to set it as it both simplifies what ACLs you must create and protects against traffic not going to a server.

Backend Management

The backend is the second piece of a standard group – dispatching requests to a set of servers. The clients who are to be dispatched must arrive from a Frontend.

The options within the Backend are identical to the Group Management options at the start of chapter 3.

ACL Management

ACL's are only required if you are using Frontends and Backends in an advanced setup, and they are used to determine to which backend a certain client arriving at a frontend will go.

This allows general concepts like having a set of servers for serving images and static content and another for dynamic content like PHP.

Current ACLs

This is a listing of the currently setup ACLs and the Predefined Defaults which exist inside the balancer.

Using the icons on the right hand side of the Your ACLs section you can edit and/or delete an ACL. Editing an ACL will automatically apply the change to any frontends utilising it.

It is important to understand that every ACL you create is always a Boolean value – it will always either be true or false. For example, is the client requesting /images/ - true. Is the clients destination port 80 – false.

Add / Edit an ACL

When adding an ACL ensure to give it a logical name as multi-ACL setups can become confusing. Snapt's standard is to name ACL's in a way that they describe their use, for example, "isImages" might refer to the path /images/.

Match Criterion

This is what the ACL is going to base its true or false evaluation on. There are many options available for advanced ACL creation here but the primary use cases are detailed below.

- dst_port – this matches based on the destination port number. For example, you could have a group listening on port 80 and 81. A value of 80 here would match clients coming in on port 80.
- src – the source IP address of the user.
- dst – the destination IP address the user connected two.
- hdr_dom(host) – the domain name in the host header. This allows you to specify a certain web domain, for example test.snapt-ui.com.
- path_beg – matches the beginning of a URL – starting after the domain. For example setting /images/ would match the following – http://test.snaptui.com/images/example/test.png
- path_end – match the end of a URL. For example, .jpg would match http://test.snapt-ui.com/images/example.jpg

Match Operator

The match operator can be "Plain" or "=". Leaving it as plain means it will match the string if it is anywhere inside the request. The = operator means it must be **exactly** what you have specified.

Value (x)

This is the value you wish to match against. For "src" it is the IP address, for "path_beg" it is the string in the beginning of the path.

Flags

Flags can be set to alter certain behaviours. Setting –i will allow it to ignore the case in the string which is typically a good idea.

Attaching ACLs

Now that you have created an ACL you can go back to Frontend Management and use the Attach an ACL feature to choose which backend should receive the request when a TRUE response is received from the ACL.

Chapter 4: Standard Operation

Now that you have your load balancer configured the next step is the control and maintenance of it.

Majority of these standard functions are available on the Balancer -> Balancer Dashboard menu option.

Here you will see several tabs, including the following –

Dashboard

This dashboard will optionally show if you enable google charts and the balancer is running.

The first three charts are gauges designed to show your current load. These are all percentage based and the Connections % is based off your chosen maximum connections value.

The sessions and throughput line charts are drawn from your chosen logging intervals and display a rough guide of your traffic flow. You can run a full report under the Reports menu item.

The IP location map has several requirements to work, which it will detail for you. Once functioning it will show you a country activity map, illustrating where your clients are currently browsing from.

Overview

The overview tab is also the main program control tab for starting and stopping the load balancer.

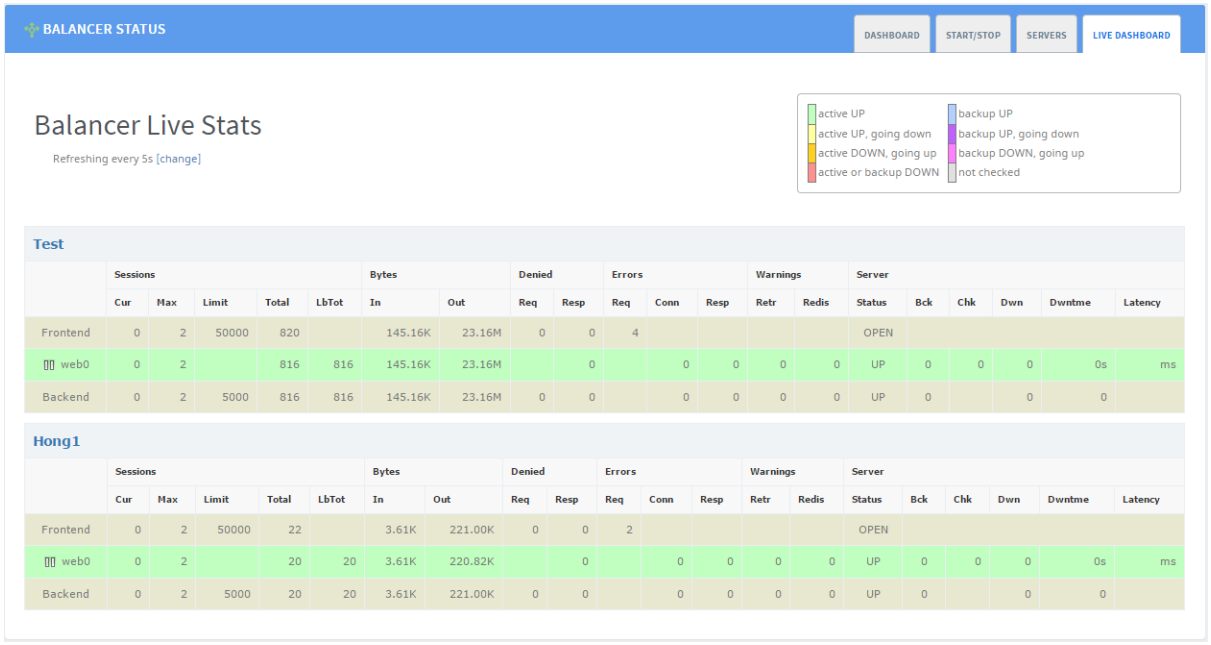
You can also see a summary status of your setup with several statistics per group.

Server Overview

This tab is a quick snapshot of your groups and the servers within them. Green statuses are the goal here, orange indicates a warning and red a critical problem. For monitoring we advise using the Full Screen menu options popups.

Live Dashboard

This is the core component of live monitoring for your load balancer. It gives you all the critical statistics and statuses of your servers, and updates automatically every 10 seconds.



Chapter 5: Reporting

A key component to your Snapt Balancer installation is the ability to draw reports, and if configured in the previous steps (under Balancer -> Configuration -> Logging Options) it will be possible to run full reports at any time through the Reports menu.

Clicking on the main Reports button will take you to an overview of all your available reports.

The Balancer will introduce 4 new reports to this module –

Proxy Throughput Daily Report – this will give you daily detailed throughput statistics for an entire group or frontend/backend. This is the most commonly used report.

Proxy Throughput Report – this will give you detailed throughput statistics for an entire group or frontend/backend. This is the most commonly used report.

Server Throughput Report – this will allow you to see individual server statistics, within a group.

Server Error Report – a detailed error log describing when a server has failed to respond, or we have had trouble connecting and so on.