



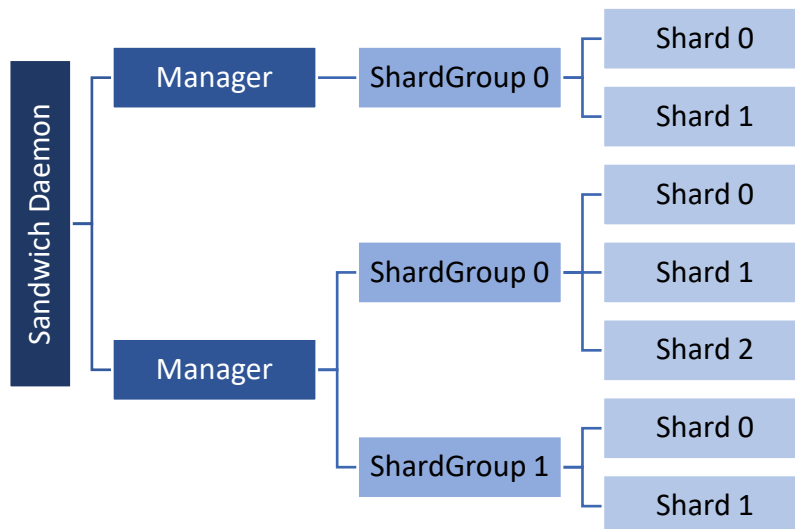
# Contents

|                           |   |
|---------------------------|---|
| INTRODUCTION              | 2 |
| DEPLOYMENT                | 2 |
| DEBUGGING SANDWICH ISSUES | 3 |
| Error Messages:           | 3 |
| Scenarios:                | 4 |
| USING THE DASHBOARD       | 4 |
| Analytics Tab             | 4 |
| Managers Tab              | 5 |
| New Manager Dialogue      | 6 |
| New ShardGroup dialogue   | 7 |
| Daemon Settings Tab       | 7 |

## Introduction

Sandwich Daemon acts as a producer microservice that connects to discord's gateway. It handles event processing, caching, and dispatching events to consumers. Sandwich Daemon allows you to manage bots (managers) by configuring a YAML file or by using a web dashboard.

Managers are the individual bots which in turn have their own Shard Groups which then contain their specific Shards.



The hierarchy model is useful as it allows for a centralised solution to managing multiple bots however can still be used for a single bot if you would want control over how your bot interacts with your gateway. The library is scalable and light and instead of using multiple processes for multiple different bots or clusters, you can use a single process for it all. By offloading caching to Redis, it also allows for a centralised place for all your caches which is useful when making a dashboard.

By implementing Shard Groups, it allows for more control over how you deploy Shards to your bot allowing you to do rolling restarts meaning you can start up new shards and kill off old shards when the new Shard Group is deployed for seamless scaling.

## Deployment



Figure 1: Recommended deployment steps

Sandwich Daemon is easy to deploy and there is a docker-compose.yml included that can be used to deploy the other services that it uses. When using Redis, it is recommended you disable saving by

using `save ""` as it will save the cache in a file which could cause excessive writes. Be aware the ports for Redis and the argument passed to nats-streaming with the argument `-p`.

Once you have deployed STAN and Redis, you are able to now start up the Daemon. If you are using other 2nd party services such as Errorly and RestTunnel, start these before Sandwich Daemon.

Once you have started up Sandwich Daemon, it will load the default config then will attempt to start up Managers if they have `AutoStart` set to true. When loading the configuration, it will attempt to mitigate any small configuration issues and either fix them or create an error when it starts up.

## Debugging Sandwich Issues

When starting up Sandwich, if it has errors it will not panic if you have `ErrOnConfigurationFailure` disabled which is configured in the source code and on by default.

### Error Messages:

- Load Configuration ReadFile – It is likely the file does not exist refer to the error specified.
- Load Configuration unmarshal – The file exists however is not valid yaml.
- Load Configuration normalize – When this occurs, there is a misconfiguration. The specific error is provided with the message.
- Manager Open ... – When this occurs, it is likely it is unable to connect to Redis or STAN. The specific error is provided.
- Failed to start up manager – This is a general error message which has an error provided with it.
- PublishEvent Marshal – The data attempted to be published could not be marshalled. This is a library problem.
- PublishEvent Publish – It is likely there was an issue with STAN
- PublishEvent Publish: No active stanClient. It is unlikely this will occur as an error is raised when it is unable to connect to STAN.
- Get Gateway FetchJSON – It is likely a network error. If you are using RestTunnel, it may be to blame so look at its logs.
- Failed to parse RestTunnel alive response – RestTunnel sent an unexpected response it could be the schema changed and you may have to update RestTunnel/Sandwich Daemon.
- Failed to connect to RestTunnel – RestTunnel is not running or the URL used was wrong. Ensure you include `http://` if you have not.
- Failed to parse RestTunnel URL – The RestTunnel URL specified was not valid.
- Failed to dial – This is due to a network error.
- Failed to read message – This is a general error message and a more detailed error is provided with it.
- Failed to send heartbeat in response to gateway – The gateway did not send a heartbeat ACK in time. If you are getting this often, it could be that your `MaxHeartbeatFailures` is too low. It is recommended to be 5.
- Failed invalid session from gateway – The gateway sent an invalid session error. This usually is not too bad but could occur if using multiple Sandwich Daemon processes on the same bot token.

- Gateway sent unknown packet – This is not too much of an issue and is only a WARN. It simply means the gateway does not support the packet op sent.
- Failed dispatching events – An issue handling an event occurred. A more specific error will be provided.
- We have encountered an error whilst in the same connection – An error with the gateway had occurred and it will attempt to reconnect
- Ran out of retries whilst connecting. Attempting to reconnect client – When reconnecting, it failed to reconnect too many times which is defined in bot > retires.

#### Scenarios:

- Managers are not starting – It is likely there are no managers that have AutoStart enabled.
- Managers are not being made – It can be the case where it had an issue creating the Manager or multiple managers have the same Identifier. If this is the case, there will be an error in console saying there are duplicate identifiers.
- The REST HTTP Interface is not enabled – You have HTTP disabled in the configuration. Whilst the HTTP is disabled, it will still show the dashboard however an error will show. You will have to restart Sandwich Daemon to fix this.
- The dashboard is not showing – You likely do not have the host set properly. It is recommended you set it to 127.0.0.1 so the dashboard is not public for others to interact with.

## Using the Dashboard

### Analytics Tab

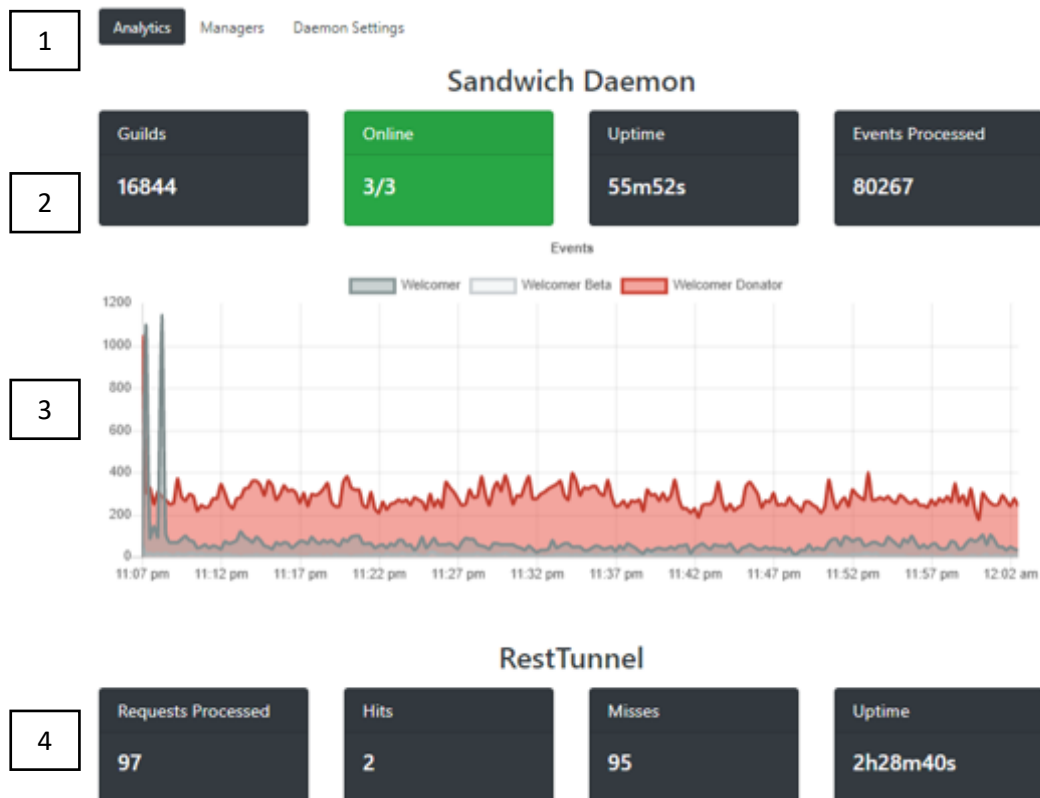


Figure 2: Analytics Section

1. **Tabs:** Allows you to change the different sections on the dashboard.
2. **Analytics:** Shows different analytics for the Daemon such as uptime, total events, online shard groups and guilds stored in redis.
3. **Event graph:** The event graph shows a rundown the events processed over a duration of time
4. **RestTunnel Analytics:** If RestTunnel was detected, its analytics will show below.

## Managers Tab

Analytics **Managers** Daemon Settings

1 **Create Manager**

2 **Status** Settings

Shard Groups 0

Average Latency ms

Sessions 0/0 (0)

3 **Scale Manager** 4 **Refresh Gateway** 5 **Delete Manager** 6 **Restart Manager**

7 manager open verify redis: dial tcp 127.0.0.1:6379: connect: connection refused

Error Welcomer Beta

8 **Starting** ShardGroup 0

9 **Stop ShardGroup**

10 Status

11 Uptime 2m45s

12 Shards 8/8

13 Average Latency -8163 ms

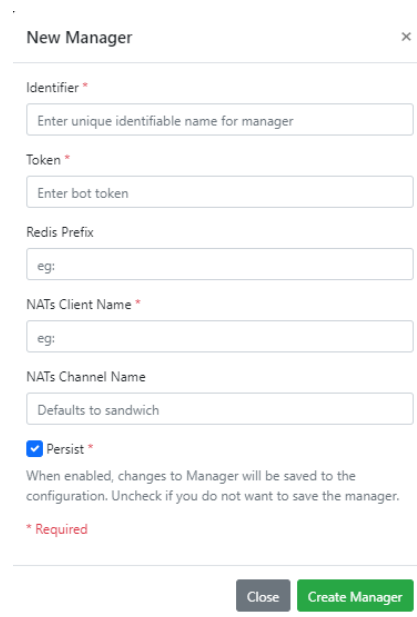
You are launching this shardgroup with an unnecessarily large number of shards. This may cause the shardgroup to take longer to load.

| Shard | Status   | RTT Latency |
|-------|----------|-------------|
| 0     | Ready    | 106ms       |
| 1     | Ready    | 116ms       |
| 2     | Ready    | 109ms       |
| 3     | Ready    | 104ms       |
| 4     | Ready    | -41251ms    |
| 5     | Starting | 0ms         |
| 6     | Idle     | 0ms         |
| 7     | Idle     | 0ms         |

Figure 3: Manager Section

1. **Create Manager Dialogue:** When clicked, allows you to create a new Manager. Refer below for the different settings within.
2. **Manager Tabs:** Allows you to view the Status of ShardGroups and the settings for the Manager.
3. **Scale Manager:** Allows you to create a new ShardGroup. Refer below for the different settings within.
4. **Refresh Gateway:** Requests the Sessions are rechecked in the event of a desync.
5. **Delete Manager:** Will stop all ShardGroups and will remove the Manager from the configuration.
6. **Restart Manager:** Will stop all ShardGroups and restart the Manager.
7. **Error Message:** If a ShardGroup Fails, the error will show.
8. **Stop ShardGroup:** Stops the currently running ShardGroup.
9. **Shard Status:** This chart colour codes the status of the current shards in the ShardGroup.
10. **Shards:** This will show the shard count and the shard count the ShardGroup uses.
11. **Average Latency:** This shows the average latency of all Shards that are not Idle.
12. **Excessive Shard Error:** This error will show if you use an excessive number of shards.
13. **Shard Statuses:** This shows the Shards; their status and their latency.

## New Manager Dialogue



**New Manager** ×

Identifier \*  
Enter unique identifiable name for manager

Token \*  
Enter bot token

Redis Prefix  
eg:

NATs Client Name \*  
eg:

NATs Channel Name  
Defaults to sandwich

☒ Persist \*  
When enabled, changes to Manager will be saved to the configuration. Uncheck if you do not want to save the manager.

\* Required

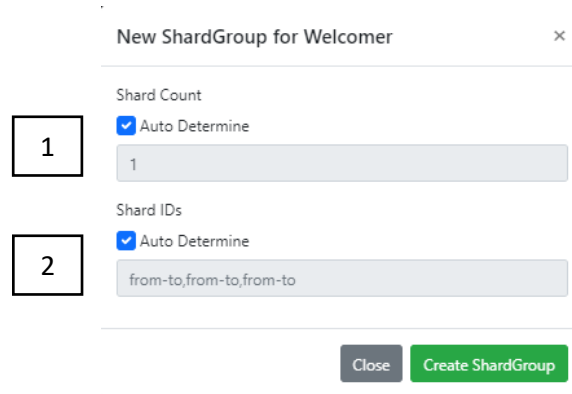
Close Create Manager

Figure 4: New Manager dialogue

1. **Identifier:** Shard identifier. Ensure this is not the same as another Manager and is unique.
2. **Bot Token:** Enter the token for the manager.
3. **Redis Prefix:** Enter the prefix that you would want for the manager. It is highly recommended this is not the same as another manager.
4. **NATs Client Name:** Client name to use with NATs/STAN. It is highly recommended this is not the same as another manager.

5. **NATs Channel Name:** The channel you want the channel to use. If not specified it will use the default one defined by the daemon. Unless you want to separate this managers events from what other consumers will process, keep this empty.

### New ShardGroup dialogue

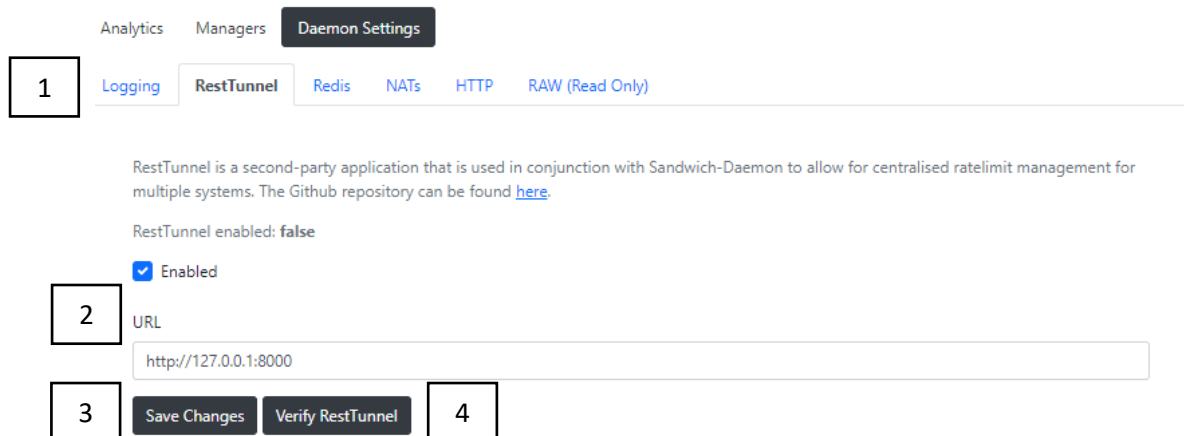


The image shows a 'New ShardGroup for Welcomer' dialog box. It has two main sections. The first section, labeled '1', is 'Shard Count' and contains a checked 'Auto Determine' checkbox and a text input field with the value '1'. The second section, labeled '2', is 'Shard IDs' and contains a checked 'Auto Determine' checkbox and a text input field with the placeholder text 'from-to,from-to,from-to'. At the bottom right are 'Close' and 'Create ShardGroup' buttons.

Figure 5: New ShardGroup dialogue

1. **Shard Count:** If Auto Determine is on, it will use what is recommended by the /gateway/bot else it will use whatever you provide.
2. **Shard IDs:** If Auto Determine is on, it will use as many shards as it can whilst respecting cluster ids aswell. If it is not on, you can provide your own rules. It is in the format to-from and is separated by , and also allows single numbers meaning it is possible to do 0,2-8 to start shards [0,2,3,4,5,6,7,8]. If you are using custom shard ids, it is recommended you also define a custom shard count.

### Daemon Settings Tab



The image shows the 'Daemon Settings' tab in a web interface. At the top are tabs for 'Analytics', 'Managers', and 'Daemon Settings'. Below these are sub-tabs: 'Logging', 'RestTunnel' (selected), 'Redis', 'NATs', 'HTTP', and 'RAW (Read Only)'. A box labeled '1' points to the 'RestTunnel' sub-tab. Below the sub-tabs, there is a description of RestTunnel and a status 'RestTunnel enabled: false'. A checkbox labeled '2' is checked and labeled 'Enabled'. Below this is a 'URL' label and a text input field containing 'http://127.0.0.1:8000'. At the bottom are two buttons: 'Save Changes' (labeled '3') and 'Verify RestTunnel' (labeled '4').

Figure 6: Daemon Settings Section

1. **Settings Tabs:** Selects the separate groups in settings.
2. **Configuration:** This is where the configuration is. Most forms that are not self-explanatory will have a description on what it does.
3. **Save Settings:** Saves the current changes to the Daemon.

4. **Verify RestTunnel:** If it is not automatically checked, asks the Daemon to check if it can see RestTunnel.

## Configuring Sandwich Managers

### General

- **AutoStart:** When AutoStart is enabled, it will attempt to create and start up a ShardGroup when Sandwich Daemon has started.
- **Persist:** When enabled, any changes to the manager configuration will be stored in the configuration file. Disable persist to temporarily make a Manager.
- **Identifier:** Identifier is the name of the manager that is used internally and referenced in event packets on STAN.

### Caching

- **Store Mutuals:** When enabled, guild ids of guilds the member is on is stored on Redis under the prefix {prefix}:mutual:{user} and is a set. **You require guild chunking to be enabled.**

### Events

- **Ignore Bots:** When enabled, MESSAGE\_CREATE events will not be sent to consumers if the author is a bot.
- **Check Prefixes:** If enabled, consumers will only receive MESSAGE\_CREATE events if it starts with a predefined prefix which is determined by a HGET on {prefix}:prefix with the guild id as a key.
- **Event Blacklist:** Events in the event blacklist will be completely ignored. It is recommended as fallback if you cannot disable it with intents.
- **Produce Blacklist:** Events in the produce blacklist will be parsed, cached and everything however will not be sent to consumers. This is useful if the consumers do not necessarily do anything with the event but its data is still important to the bots state.

### Sharding

- **Cluster Count/ID:** Clustering is useful if you are AutoSharding and are running Sandwich Daemon on multiple computers. If you have only one daemon, use a cluster count of 1 and cluster id of 0.