# Monitoring and Troubleshooting

## **Statsd reporting**

Zuul comes with support for the statsd protocol, when enabled and configured (see below), the Zuul scheduler will emit raw metrics to a statsd receiver which let you in turn generate nice graphics.

### **Configuration**

Statsd support uses the statsd python module. Note that support is optional and Zuul will start without the statsd python module present.

Configuration is in the statsd section of zuul.conf.

### **Metrics**

These metrics are emitted by the Zuul Scheduler:

**zuul.event.<driver>.<type>*(counter)***

Zuul will report counters for each type of event it receives from each of its configured drivers.

**zuul.connection.<connection>**

Holds metrics specific to connections. This hierarchy includes:

**zuul.connection.<connection>.cache.data\_size\_compressed*(gauge)***

The number of bytes stored in ZooKeeper for all items in this connection’s change cache.

**zuul.connection.<connection>.cache.data\_size\_uncompressed*(gauge)***

The number of bytes required to for the change cache (the decompressed value of data\_size\_compressed).

**zuul.tenant.<tenant>.event\_enqueue\_processing\_time*(timer)***

A timer metric reporting the time from when the scheduler receives a trigger event from a driver until the corresponding item is enqueued in a pipeline. This measures the performance of the scheduler in dispatching events.

**zuul.tenant.<tenant>.event\_enqueue\_time*(timer)***

A timer metric reporting the time from when a trigger event was received from the remote system to when the corresponding item is enqueued in a pipeline. This includes zuul.tenant.<tenant>.event\_enqueue\_processing\_time and any driver-specific pre-processing of the event.

**zuul.tenant.<tenant>.management\_events*(gauge)***

The size of the tenant’s management event queue.

## **Prometheus monitoring**

Zuul comes with support to start a prometheus metric server to be added as prometheus’s target.

### **Configuration**

To enable the service, set the prometheus\_port in a service section of zuul.conf. For example setting scheduler.prometheus\_port to 9091 starts a HTTP server to expose metrics to a prometheus services at: http://scheduler:9091/metrics

### **Metrics**

These metrics are exposed by default:

**process\_virtual\_memory\_bytes*(gauge)***

**process\_resident\_memory\_bytes*(gauge)***

**process\_open\_fds*(gauge)***

**process\_start\_time\_seconds*(gauge)***

**process\_cpu\_seconds\_total*(counter)***

On web servers the following additional metrics are exposed:

**web\_threadpool\_idle*(gauge)***

The number of idle workers in the thread pool.

**web\_threadpool\_queue*(gauge)***

The number of requests queued for thread pool workers.

**web\_streamers*(gauge)***

The number of log streamers currently in operation.

## **Tracing**

Zuul includes support for [distributed tracing](https://zuul-ci.org/docs/zuul/latest/tracing.html#distributed-tracing) as described by the OpenTelemetry project. This allows operators (and potentially users) to visualize the progress of events and queue items through the various Zuul components as an aid to debugging.

OpenTelemetry defines several observability signals such as traces, metrics, and logs. Zuul uses other systems for metrics and logs; only traces are exported via OpenTelemetry.

Zuul supports the OpenTelemetry Protocol (OTLP) for exporting traces. Many observability systems support receiving traces via OTLP (including Jaeger tracing).

## **Thread Dumps and Profiling**

If you send a SIGUSR2 to one of the daemon processes, it will dump a stack trace for each running thread into its debug log. It is written under the log bucket zuul.stack\_dump. This is useful for tracking down deadlock or otherwise slow threads:

When yappi (Yet Another Python Profiler) is available, additional functions’ and threads’ stats are emitted as well. The first SIGUSR2 will enable yappi, on the second SIGUSR2 it dumps the information collected, resets all yappi state and stops profiling. This is to minimize the impact of yappi on a running system.

## **Zuul Maintainers**

The Zuul project is self-governed.

Decisions regarding the project are made by the Zuul Maintainers. They are a team of people who are familiar with the project as a whole and act as stewards of the project. They have the right to approve or reject proposed changes to the codebase, as well as make other decisions regarding the project.

The Maintainers are expected to be familiar with the source code, design, operation, and usage of all parts of the Zuul project. When acting in their capacity as Maintainers, they are expected to consider not only their own needs, but those of the entire community.

Changes to the code, documentation, website, and other project resources held in version control repositories are reviewed and approved by the Maintainers. In general, approval is sought from at least two maintainers before merging a change, but fewer or more reviews may be warranted depending on the change. Factors to consider when reviewing are the complexity of the change, whether it is in accordance with the project design, and whether additional project participants with subject matter expertise should review the change. Maintainers may also reject changes outright, but this is expected to be used sparingly in favor of (or in the process of) redirecting effort toward efforts which can achieve consensus.

The purpose of this review process is two-fold: first, to ensure that changes to the project meet sufficiently high standards so that they improve the project, contribute to furthering its goals, and do not introduce regressive behavior or make the project more difficult to support in the future. Secondly, and just as important, the process also ensures that contributors are aware of the changes to the project. In a distributed environment, reviews are an important part of our collaborative process.

Project decisions other than those involving on-line review are discussed on the project mailing list. Anyone is welcome and encouraged to participate in these discussions so that input from the broader community is received. As the authority, Maintainers should strive to achieve consensus on any such decisions.

Changes to the membership of the Maintainers are decided by consensus of the existing maintainers, however, due to their sensitivity, these discussions should occur via private communication among the maintainers under the direction of the Project Lead.

A large group of Maintainers is important for the health of the project, therefore contributors are encouraged to become involved in all facets of maintenance of the project as part of the process of becoming a Maintainer. Existing Maintainers are expected to encourage new members. There are no artificial limits of the number of Maintainers. The Project Lead will assist any contributor who wishes for guidance in becoming a Maintainer.

Current Zuul Maintainers:

| **Name** | **Matrix ID** |
| --- | --- |
| Clark Boylan | Clark (@clarkb:matrix.org) |
| Felix Edel | felixedel (@felixedel:matrix.org) |
| Ian Wienand | ianw (@iwienand:matrix.org) |
| James E. Blair | corvus (@jim:acmegating.com) |
| Jens Harbott |  |
| Jeremy Stanley | fungi (@fungicide:matrix.org) |
| Monty Taylor | mordred (@mordred:inaugust.com) |
| Simon Westphahl | swest (@westphahl:matrix.org) |
| Tobias Henkel | tobiash (@tobias.henkel:matrix.org) |
| Tristan Cacqueray | tristanC (@tristanc\_:matrix.org) |

## **Zuul Project Lead**

The Maintainers elect a Project Lead to articulate the overall direction of the project and promote consistency among the different areas and aspects of the project. The Project Lead does not have extra rights beyond those of the Maintainers, but does have extra responsibilities. The Project Lead must pay particular attention to the overall design and direction of the project, ensure that Maintainers and other contributors are familiar with that design, and facilitate achieving consensus on difficult issues.

If the project is unable to achieve consensus on an issue, the Project Lead may poll the Maintainers on the issue, and in the case of a tie, the vote of the Project Lead will be the tie-breaker.

The Project Lead is elected to a term of one year. The election process shall be a Condorcet election and the candidates shall be self-nominated from among the existing Maintainers.

**The Project Lead is James E. Blair (term expires 2024-01-16).**

## **Zuul-Jobs Maintainers**

The zuul-jobs and zuul-base-jobs repositories contain a standard library of reusable job components which are designed to be used in a wide variety of situations.

Changes to these repositories require consideriation of the various environments in which the jobs may be used as well as policies which promote the consistency and stability of the components therein, but not necessarily the full scope of Zuul development. To that end, approval rights for changes to these repositories are granted to both the Zuul Maintainers and an additional group known as the Zuul-Jobs Maintainers.

Current Zuul-Jobs Maintainers (in addition to Zuul Maintainers):

| **Name** | **Matrix ID** |
| --- | --- |
| Andreas Jaeger |  |
| Mohammed Naser |  |
| Albin Vass | avass (@avass:vassast.org) |