+++ title = “Configure Grafana Live” description = “Grafana Live configuration guide” keywords = [“Grafana”, “live”, “guide”, “websocket”] weight = 120 +++

# Configure Grafana Live

Grafana Live is enabled by default. In Grafana v8.0, it has a strict default for a maximum number of connections per Grafana server instance.

## Max number of connections

Grafana Live uses persistent connections (WebSocket at the moment) to deliver real-time updates to clients.

WebSocket is a persistent connection that starts with an HTTP Upgrade request (using the same HTTP port as the rest of Grafana) and then switches to a TCP mode where WebSocket frames can travel in both directions between a client and a server. Each logged-in user opens a WebSocket connection – one per browser tab.

The number of maximum WebSocket connections users can establish with Grafana is limited to 100 by default. See [max_connections]({{< relref “../administration/configuration.md#max_connections” >}}) option.

In case you want to increase this limit, ensure that your server and infrastructure allow handling more connections. The following sections discuss several common problems which could happen when managing persistent connections, in particular WebSocket connections.

## Request origin check

To avoid hijacking of WebSocket connection Grafana Live checks the Origin request header sent by a client in an HTTP Upgrade request. Requests without Origin header pass through without any origin check.

By default, Live accepts connections with Origin header that matches configured [root_url]({{< relref “../administration/configuration.md#root_url” >}}) (which is a public Grafana URL).

It is possible to provide a list of additional origin patterns to allow WebSocket connections from. This can be achieved using the [allowed_origins]({{< relref “../administration/configuration.md#allowed_origins” >}}) option of Grafana Live configuration.

### Resource usage

Each persistent connection costs some memory on a server. Typically, this should be about 50 KB per connection at this moment. Thus a server with 1 GB RAM is expected to handle about 20k connections max. Each active

connection consumes additional CPU resources since the client and server send PING/PONG frames to each other to maintain a connection.

Using the streaming functionality results in additional CPU usage. The exact CPU resource utilization can be hard to estimate as it heavily depends on the Grafana Live usage pattern.

### Open file limit

Each WebSocket connection costs a file descriptor on a server machine where Grafana runs. Most operating systems have a quite low default limit for the maximum number of descriptors that process can open.

To look at the current limit on Unix run:

```
ulimit -n
```

On a Linux system, you can also check out the current limits for a running process with:

```
cat /proc/<PROCESS_PID>/limits
```

The open files limit shows approximately how many user connections your server can currently handle.

To increase this limit, refer to these instructionsfor popular operating systems.

### Ephemeral port exhaustion

Ephemeral port exhaustion problem can happen between your load balancer (or reverse proxy) software and Grafana server. For example, when you load balance requests/connections between different Grafana instances. If you connect directly to a single Grafana server instance, then you should not come across this issue.

The problem arises because each TCP connection uniquely identified in the OS by the 4-part-tuple:

```
source ip | source port | destination ip | destination port
```

By default, on load balancer/server boundary you are limited to 65535 possible variants. But actually, due to some OS limits (for example on Unix available ports defined in `ip_local_port_range` sysctl parameter) and sockets in TIME_WAIT state, the number is even less.

In order to eliminate a problem you can:

- Increase the ephemeral port range by tuning `ip_local_port_range` kernel option.
- Deploy more Grafana server instances to load balance across.
- Deploy more load balancer instances.
- Use virtual network interfaces.

### WebSocket and proxies

Not all proxies can transparently proxy WebSocket connections by default. For example, if you are using Nginx before Grafana you need to configure WebSocket proxy like this:

```
http {
    map $http_upgrade $connection_upgrade {
        default upgrade;
        '' close;
    }

    upstream grafana {
        server 127.0.0.1:3000;
    }

    server {
        listen 8000;

        location / {
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection $connection_upgrade;
            proxy_set_header Host $http_host;
            proxy_pass http://grafana;
        }
    }
}
```

See the Nginx blog on their website for more information. Also, refer to your load balancer/reverse proxy documentation to find out more information on dealing with WebSocket connections.

Some corporate proxies can remove headers required to properly establish a WebSocket connection. In this case, you should tune intermediate proxies to not remove required headers. However, the better option is to use Grafana with TLS. Now WebSocket connection will inherit TLS and thus must be handled transparently by proxies.

Proxies like Nginx and Envoy have default limits on maximum number of connections which can be established. Make sure you have a reasonable limit for max number of incoming and outgoing connections in your proxy configuration.