## Important metrics to gather

By analyzing the request, i realized that the specific tasks to do are finding the important metrics for the use case and giving some details why they could be important, giving some generalization on the proxy software and ssl offloading main components, how and if they can be monitored, and then giving a real scenario, based on this generalization. In the end, i will touch upon some challenges we could face about this issue. Also, it is supposed that the proxy server and ssl offloading is already configured and set up.

Two important components of a proxy server(which could include ssl offloading) are the frontends and backends. The frontends are the endpoints or urls that clients use to access api-s and similar, and backends are the ip-s of the servers responsible for delivering the required content. In between them may be some other services such as ssl offloading ( termination where data coming from the clients through the frontends is decrypted and then sent unencrypted to the backend servers, or bridging where data is decrypted to perform extra cheks for malware and similar before encrypting it again and then sending it to backends). These components have parameters such as max connections per second,max connection time, max response time worth saving as metrics such as number of connections at any given time, uptime, downtime, bytes in, bytes out, request queue, error rate for requests, error rate for responses, denied requests, denied responses also global metrics such as number of active backends, backup backends. These metrics help analyze the performance of each backend or frontend related to the overall performance and stability of the target service such as ssl offloading.

Since ssl offloading apart from network metrics mentioned above, because of it's nature to give a heavy load in cpu as a result of complex algorithms, and a heavy load on ram usage to have the fastest way to handle data encryption and decryption in multiple concurrent connections, it is important also to have detailed metrics of these two components i.e detailed usage of cpu and ram. Also the overall network usage. All of these metrics help analyze the potential bottlenecks in ssl offloading in the scenario given, optimize the system for that metric, and mitigate any possible errors reported by the monitoring system.

## How to monitor these metrics

Apart from a monitoring system gathering the data, it would be convenient to have a system to visualize and categorize such data for further analysis. Also the underlying software for proxy services should have some kind of integration with the monitoring system in order to

gather all the required data more effectively by utilizing the own proxy software stats provider if possibly, than just monitoring software which gathers data unaware of the proxy software used. Another important process would be a notification system for alerts and errors that may occur to help analyze and solve the problems quickly. Since the hardware mentioned in the scenario is exclusively used for ssl offloading and proxy services, than it would be necessary having extra hardware configurations for the monitoring system.

How to choose the optimal software for our scenario? If we want all these different configurations for monitoring, alerting and possible integration with the proxy services, than the software used for this should be very extensible and featureful. A very popular soulution that ticks most of these is using Prometheus. Prometheus is extensible supporting client integrations for dozens of programming languages and also available integrations for a lot of popular services. Apart from integrations with various proxy services such as nginx, haproxy, traefik etc, it also has support for monitoring their docerized forms. For ram, cpu and network usage metrics it is better to use node exporter for prometheus. For proxy service metrics ( especially individual metrics for frontends and backends) there can be used the appropriate exporter, for instance if the proxy software is haproxy, the exporter used in this case is haproxy exporter and so on. Prometheus could be integrated with Graphana for visualizations of various metrics of ( ram, cpu, processes, frontends and backends with their respective metrics such as number of requests per second, error rate, requests denied, time of request, time of connection, time of response, etc). Fortunately Prometheus supports an alerting system through AlertManager. The AlertManager handles alerts sent by client applications such as the Prometheus server. It takes care of deduplicating, grouping, and routing them to the correct receiver integration such as email, PagerDuty, or OpsGenie ( also third party integrations for sms, aws, google cloud etc). It also takes care of silencing and inhibition of alerts.

## Challenges

Still there are some challenges before moving on using Prometheus. Prometheus is open source and maintaned mostly by community so it doesn't have the same accountability as a commercial product for stability. There are other very popular open source solutions such as Zabbix or Nagios which offer paid plans for commercial use of their products and such have better support for their clients that use these software. There is also SolarWinds which is not

open source, and has a quota based payment system, but has very good support and integration with various services including native azure, active directory, aws, centos monitoring. There is also the financial cost and human resources of setting up and configuring these services ( also integrating them with the current proxy services), and manpower to monitor, report and analyze the data.