sysbee
Serving your uptime

System Bee d.o.o.
www.sysbee.net
info@sysbee.net

Infrastructure assessment

Description of the assessment process

Example of an assessment deliverable

Table of contents

The infrastructure assessment process

The infrastructure assessment process starts with a call with the customer where we establish the scope of the assessment and estimate the work hours required to finish the assessment. The assessment costs 80€ per hour, and in most cases, we require 5 hours to complete it, depending on the complexity of the infrastructure. However, this does not mean the assessment process will be completed in a single day, as we inspect the systems at various intervals and gather data live or through longer term monitoring (usually one week, but can be longer if longer term monitoring is agreed upon).

The process is almost exclusively read only as we do not make any changes on the current live infrastructure, but rather observe its settings and function in a live environment. The idea is to provide suggestions to improve performance, security and give the customer a better insight into his current infrastructure.

The assessment process covers the follow following points on Linux based infrastructures:

- Current infrastructure diagram and capacity
- OS level inspection
- Service level inspection
- Performance tunables inspection (both OS and service level)
- Database service inspection, and performance tunables
- Security aspect of infrastructure, including firewalls, certificates, service setup, IDS
- Backup policies review
- Web front-end diagnostics and suggestions (typically 1-2 websites on infrastructure)
- Domain name mappings and DNS inspection
- To some degree application level performance impact analysis on current infrastructure
- Dedicated server hardware tunables (if applicable)
- New infrastructure proposal (if applicable)
- Cost saving analysis (if applicable)

The process does not cover the following:

- Application level inspection
- Application level debugging
- Applying any system changes and optimizations suggested within assessment deliverable
- Security hardening implementation
- Internal business and development processes review
- MS Windows or BSD/Unix alike systems

During the assessment process we can offer performance monitoring and graphing as an additional step. This is not a mandatory step, however it gives us a better insight into daily server operations, especially with live systems where bottlenecks might not be apparent during a short window of live inspection.

With longer term monitoring we install a single package on servers that would be used as monitoring probes for system performance data analysis and trends observations.

We do not gather any customer related data, codebase or user uploaded materials, and the data we gather is anonymized and it is not leaving EU territory.

We usually gather data for at least 1 week prior to finishing deliverables, but we can shorten or prolong this period based on customer requirements. This process takes no extra charge and monitoring daemon is removed from your systems by the end of assessment.

Deliverables

As a deliverable, the customer will get a comprehensive infrastructure assessment whitepaper. The whitepaper will include a detailed description of the current infrastructure status, as well as optimization tips and guides which can then be implemented by the customer, by their current integrator or we can agree upon applying suggested fixes outside the scope of this assessment.

We are reserving the right to decline any modifications if we deem that modifying the current live environment will affect system stability or performance, or in cases where a newly redesigned infrastructure would be a better choice. If that’s the case, we will include this conclusion in the whitepaper as well.

In the following section we’ve included an anonymized analysis we performed for one of our customers. It should give you a better idea of what the deliverables look like, what we usually analyse and how.

Server.com

Infrastructure assessment

Conducted by: (sysadmin)
System bee d.o.o.
(date)

Table of contents

Introduction

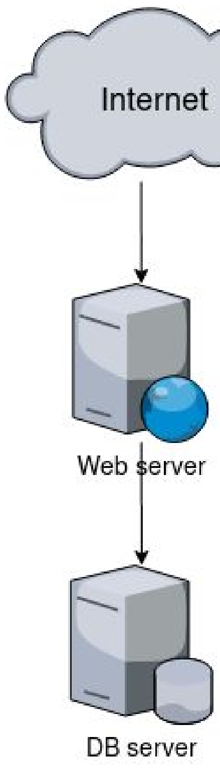
First of all, we'd like to thank you for trusting Sysbee to conduct your server assessment. As you may have already heard, we have vast experience in doing small to large hosted or cluster deployments and have a very good background on running and scaling large systems instances. We are constantly striving to improve this process and tune every conceivable server parameter so that we get the maximum performance out of it.

In this document, you will find the inspected state of your current server setup and our recommendations for both performance and security improvements.
We sincerely hope that, by following our suggestions and recommendations in this document, you will improve the performance of the Magento instance, as well as the server itself, and that you will overcome issues that you are currently experiencing.

Please take some time to go through this document, we can later schedule a call conference to discuss it in more depth if necessary.

Current infrastructure overview

Diagram



Infrastructure description

The current infrastructure consists of two KVM virtual machines. One VM is used as a web application server (with Nginx and PHP-FPM as the most important services). The other server acts as a database server (with Redis and Percona server).

Services analysis

Web based

Nginx

The web application server is running the latest stable version of Nginx (at the time of writing, this is version 1.16.1). From both performance and security standpoint, Nginx is configured very well however, we found a couple of things that can be improved:

- TLS 1.0 protocol should be disabled, as it's insecure. The server has support for newer TLS versions (1.1 and 1.2), which are supported by all popular web browsers and mobile devices. This can be achieved by adding the following directive to Nginx configuration:

```
ssl_protocols TLSv1.1 TLSv1.2;
```

- Since the Magento store at www.server.com has a valid SSL certificate, for improved security we strongly recommend that HTST (HTTP Strict Transport Security) header is added to the website's Nginx virtualhost:

```
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" preload;
```

- For improved security, Nginx should use only the most secure SSL/TLS ciphers, which at the same time offer good compatibility with modern clients:

```
ssl_ciphers
ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-G
CM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-
CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384;
ssl_prefer_server_ciphers off;
```

PHP

Most global PHP settings have sensible values. The most important limits that can prevent resource exhaustion (such as max_execution_time and max_input_time) are properly configured, but for improved performance and stability of the web application server, we recommend the following:

- PHP memory_limit should be lowered from 2 GB to 1 GB. We have found that even large Magento 2 installations (with multiple stores) can function without any issues when PHP memory_limit is set to 1 GB.

  Most resource-heavy actions are executed via PHP CLI (e.g. cron jobs, database migrations, etc.), where memory_limit can easily be overridden. Having a lower default memory_limit minimizes the impact of potentially high memory consumption for requests that were sent over HTTP protocol.

- For slight performance and stability improvement, we recommend updating a couple of OPcache directives:
  - opcache.enable_cli should be disabled as it doesn't bring significant performance improvement, but it can cause unexpected application behavior when working in CLI mode.
  - opcache.fast_shutdown should be turned off to avoid unexpected problems (such as "zend_mm_heap corrupted" errors) when PHP-FPM childs are recycled.

- Given the large codebase that Magento 2 has, increasing opcache.memory_conumption from 256 MB to 512 MB should give better performance and lesser opcache memory fragmentation.

```
opcache.enable_cli            = 0
opcache.fast_shutdow          = 0
opcache.memory_conumption     = 512
```

While assessing PHP-FPM configuration, we have noticed that child limits are set very high compared to available server resources, namely the number of CPU cores.
On the server with 16 CPU cores, the pm.max_children limit of 128 leaves the opportunity for a high context switching scenario, which will, in turn, prolong PHP execution time. We recommend lowering this limit to a more sensible limit of 60 child processes. This change also implies lowering related PHP-FPM process limits:

```
pm.max_children         = 60
pm.start_servers        = 2
pm.min_spare_servers    = 2
pm.max_spare_servers    = 5
```

Varnish

Varnish cache is currently not installed on this server. Since Magento 2 natively supports Varnish, we recommend that Varnish web accelerator is put in front of the Nginx web server. This will improve overall performance and lower the number of requests reaching PHP-FPM and therefore reduce resource usage.

You can find more information about Varnish integration in official Magento 2 documentation.

Database services

MySQL

During the assessment of the Percona server, we have noticed that the max_connections limit is reached every couple of days. This is potentially caused by the fact that the Percona server is improperly configured and is able to allocate almost double the amount (28.4 GB of available memory on the server (16 GB).
To lower theoretical memory consumption by the Percona server and at the same time improve both stability and performance, we recommend that the following changes are applied:

- local_infile should be disabled for improved security.
  query_cache_size should be increased to 256 MB. We don't recommend increasing this limit above 256 MB as larger query cache size can introduce query cache locking problems.

- for improved performance, innodb_buffer_pool_size should be greater or equal to InnoDB data size (15.8 GB) whenever possible. However, since the database server has 16 GB of memory and it's running Redis as well, we recommend that innodb_buffer_pool_size doesn't exceed 10 GB. The rest of the memory should be reserved for Redis and OS caches and buffers.

- wait_timeout and interactive_timeout should be lowered from default 28800 seconds (8 hours) to 300 seconds (5 minutes) to minimize the chance of reaching max_connections limit due to too many inactive connections.

- key_buffer_size should be lowered from 436 MB to 128 MB, since only 18% (80 MB) is actually being used.

- join_buffer_size, tmp_table_size and max_heap_table_size should be lowered to avoid high memory usage when numerous SQL queries are running concurrently.

- thread_cache_size should be increased to match the max_connections limit.

```
local-infile              = 0
query_cache_size          = 512M
innodb_buffer_pool_size   = 10G
wait_timeout              = 300
interactive_timeout       = 300
key_buffer_size           = 128M
join_buffer_size          = 1M
tmp_table_size            = 128M
max_heap_table_size       = 128M
thread_cache_size         = 100
```

Given the size of the development database, we also recommend migrating it to a dedicated virtual machine to completely separate the production and development environment. This will improve both security and performance.

If separating production and development environments is not possible, we would recommend adding an additional 8 GB of memory to the database server. With additional memory, overall performance can be improved by increasing innodb_buffer_pool_size to 16 GB and setting tmpdir="/dev/shm".
With additional memory, temporary tables that would usually be created on disk would, in this case, be created in a ramdisk, which will greatly improve performance when dealing with large temporary tables.

While inspecting the databases, we have detected two defragmented tables. By optimizing them, an estimated 40 GB of disk space could be reclaimed. This will also improve performance in situations when these tables are queried.

```
OPTIMIZE TABLE `db_1`.`sample_table_1`;
OPTIMIZE TABLE `db_2`.`sample_table_2`;
```

Redis

Production database server is also running Redis. For improved security and stability, we recommend the following changes:

- upgrading Redis from the currently installed version (3.2) to the latest stable version (5.0). Magento 2 is fully compatible with Redis 5.0.

- maxmemory limit is currently not defined, which means that Redis will use as much memory as it can. Setting maxmemory limit is highly recommended to avoid OOM (out-of-memory) situations on the server. In this particular case, maxmemory limit of 1 GB seems the most appropriate.

- maxmemory-policy is currently set to "noeviction", which means that Redis won't evict any keys when it reaches maxmemory limit or when the server runs out of free memory.

Since Magento 2 automatically sets expiration TTL on saved keys, "maxmemory-policy volatile-ttl" is the most appropriate. If Redis is upgraded to version 5.0, then we would recommend setting "maxmemory-policy volatile-lru".

- Redis databases are too frequently flushed to disk (every 5 minutes). IO performance can be improved if Redis save policy is updated to occur once per day. Since Redis contains mostly temporary data (caches and sessions), less frequent save policy shouldn't present a problem. Please note that databases will automatically be saved on disk when Redis service is gracefully restarted.

```
maxmemory 1g
maxmemory-policy volatile-ttl    # or volatile-lru if Redis >= 4.0
save 86400 1
```

System analysis

Both web and database servers have available updates. We recommend updating all packages, especially those that are from base CentOS repository, to ensure that all security and bug fixes are applied.

On the web server, we have found that the firewall is not properly configured. Both INPUT and OUTPUT iptables chains have default policy set to ACCEPT. We strongly recommend that you apply strict firewall policy to restrict unnecessary public access to specific services (e.g. rpcbind on TCP port 111 is currently exposed, which presents a security problem).

For improved security, we strongly recommend that SSH password authentication is disabled in favor of PubkeyAuthentication and that root login is restricted to whitelisted IPs. If possible, SSH access should be limited to specific IPs as well.

We also recommend disabling the UseDNS feature which doesn't really improve security, but in some cases it can introduce login delay.

We suggest that the following changes are applied to /etc/ssh/sshd_config

```
PasswordAuthentication no
PubkeyAuthentication yes
UseDNS no

# To allow root login from specific IPs or IP ranges
PermitRootLogin no
Match Address 123.123.123.123/32,234.234.234.0/24
  PermitRootLogin yes
```

We recommend rebooting both servers because they are running older Linux kernel versions. If you wish to avoid server reboots, we strongly recommend installation of KernelCare live kernel patching service.

Conclusion

Security-related suggestions mentioned in this document should be applied first. e.g. proper iptables configuration for the web server, SSH server hardening, etc.

By applying performance optimizations suggested in this document, we believe that performance and stability of key services will improve noticeably.
Since the database server is having some service impacting problems (e.g. reaching Percona server's max_connections), an occasional 404 error on server.com frontpage may be caused by memory pressure on the database server.

With current Redis configuration (without maxmemory limit and with "noeviction" policy), Magento page cache may be partially written because Redis is unable to free its memory space. This could explain why flushing Magento cache resolves the 404 error on the frontpage.