

Remiges *ServerSage*

Managed infra service

Product setup

1 Background

We wish to offer *ServerSage*, a managed infrastructure (servers, services, etc) service as a productised service (PSvc) to our customers. This means that once we have the productised service (PSvc) ready, we will have an exact set of collateral, tools and processes for

- BoM for quoting for a new lead
- Setup
- Operation
- Incident handling and escalations

This note describes what we need to do as homework to launch *ServerSage*. This “homework” is being treated as an internal asset-building project.

When this project is completed, the following outcomes will be achieved:

- We will have a detailed two level setup checklist for
 - each type of server and each type of (commonly found) service for the setup tasks. The first-level checklist will be the server/service types we deal with, and the second-level list will be the list of attributes we monitor under each.
 - Each repeatable additional service (e.g. patch application, software upgrade, data backup and restore) which can be provided as add-on PSvc with the basic monitoring service
- We will have a similar two-level training manual for imparting training to L1, L2, and L3 infra engs to monitor, drill down, and escalate for each type of server/service.
- We will have a set of one-page service descriptions, one per PSvc
- We will have a pre-sales checklist to assist our pre-sales chaps about what each PSvc includes or excludes

2 Types of servers/services

Servers:

- Linux
- Windows

Auxiliary components:

- Connectivity
- Network throughput

Services and layered products:

- Tomcat
- NGINX
- Apache HTTPd
- MySQL
- Postgres
- MS SQL Server
- MongoDB
- Oracle
- MS IIS
- MS Active Directory
- Redis
- Keycloak
- Red Hat SSO
- Red Hat JBoss
- WildFly
- Kafka
- RabbitMQ
- IBM MQ
- MS MQ

Layered products we will add later, because they are either growing in popularity or are waning in popularity:

- IBM DB/2
- Kubernetes
- VMware

3 Monitoring framework

We intend to offer our service by using a foundation of Prometheus plus agents for monitoring. We will use Grafana as our monitoring UI, plus any other visualisation tools we may need later.

All data will be stored in the database which Prometheus uses.

We will program alerts into each attribute being monitored, so that when watermarks are breached, SMS and emails will be pushed out by the system.

The client will provide infrastructure in their cloud or on-prem DC to set up and operate this monitoring stack, and will provide connectivity to the Internet to let us push out SMS and emails.

We will develop our own custom reporting UI which will generate canned reports for governance purposes, for archival and filing to accompany invoicing, and for ad hoc requests from the client.

4 Parameters to be monitored

This section will list the health parameters which need to be monitored for each server or service, together with the tools and processes needed for such monitoring.

Parameters marked **in dark red** are ones for which we will need to write agent software. Some of the agents will be really small shellscripts. We are stating, till we are proved wrong, that all the parameters listed below, which are in black, will be logged by standard agents which are readily available, and all the ones **in dark red** will need to be developed as Prometheus agents.

4.1 Linux OS

- CPU load
- RAM usage
- Swap space usage
- Number of processes running:
 - Total
 - **Specific processes**
- **Specific list of network ports where listeners are listening (netstat -lnp | grep for specific services)**
- Disk space available
- Free inodes
- **Alert messages whenever OS reboots**
- Disk I/O volumes
- Network interface traffic volumes
- **Grep for specific lines or errors automatically in dmesg**
- **Change monitor (not in Prometheus):**
 - **a daily log of versions of kernel and all applications, to track changes due to upgrades/patches**
 - **config files**
 - **Mounted file systems list**
 - **IPtables ruleset**
 - **Number of CPU cores (needed for VMs)**
 - **a file system integrity and change detection daemon, which will log all changes to the file system metadata (new files, changes to file attributes, etc)**

4.2 Windows OS

- CPU, RAM, swap, nprocesses, listeners on network ports, disk space available, free FS nodes, disk I/O volumes, network interface traffic volumes, alert when OS reboots, change monitor: **as above**

4.3 Connectivity

- Outward connection tests, with record of time taken
 - Ping: north-south and east-west:
 - latency
 - packet loss
 - jitter
 - HTTP requests (via proxy server, if proxy is active)
 - Connection to DB server(s)

4.4 Network throughput

- Test access and do bulk data transfer, measure the throughput with
 - Specific monitoring URLs set up by us on the Internet
 - East-west with the same DC

4.5 Web servers: Apache HTTPd, NGINX

- Number of connections: total and active
- Number of processes/threads
- Probe the service and measure the response time and status

4.6 Application servers: Tomcat, JBoss, MS IIS, WildFly

- Number of connections: total and active
- Number of processes/threads
- Probe the service and measure the response time and status
- JVM memory consumed (for Tomcat, JBoss, etc)
- (from logfile): time taken for each GC round, and whether partial or full GC

4.7 Database services

The set includes, for now: MySQL, MS SQL, Postgres, Oracle, and MongoDB.

- Number of connections: total and active
- Number of processes/threads
- Probe the service and measure the response time and status
- Memory consumed by server (will be DB specific)
- Change monitor (not in Prometheus):
 - Number of tables
 - Number of rows in the top N largest tables
 - Persistent data size on disk

- Provide a wrapper script to do VACUUM ANALYZE and log it (DB specific)
- Count of active locks
- Replication status:
 - When did the last snapshot replicate?
 - Status of snapshot application on replica server: OK or error
- (from logfile) Count of slow-running queries in the last minute

4.8 Directory services and cache: MS AD, Redis

- Number of connections: total and active
- Number of processes/threads
- Probe the service, measure the response time and status
- Size of data in the service (nrows in cache, etc)

4.9 Authentication services: Keycloak, RH SSO

- Number of connections: total and active
- Number of processes/threads
- Probe the service, attempt an authentication, measure the response time and status

4.10 Message bus: Kafka, RabbitMQ, MS MQ, IBM MQ

- Number of connections: total and active
- Number of messages in each queue
- Run a test producer and test consumer, measure the response time and status
- Run a test consumer for all pub-sub services, track the number of messages being seen per queue per minute

5 Action plan for product setup

This section specifies how we will develop the PSvc. This is equivalent to the WBS for the product-creation project.

We assume that we have one client, and we will deliver our PSvc bouquet in parallel with delivering service to them. The rollout can be done in chunks as structured below.

5.1 Project kick-off

1. Discuss with a client and prepare a rollout plan
2. Make an inventory of entities to support
3. Get resources allocated for running our monitoring system in their DC
4. Set up Prometheus on the server allocated
5. Test outgoing SMS and email alerts, collaborate with client to enable it

Do not actually start any monitoring agents at this stage, just set up the monitoring infrastructure. No actual useful service delivery to the client happens at this stage.

5.2 L1 support team setup

6. Set up agents for all the parameters of all the servers/services which are in black in the list above
7. Define the watermarks for each parameter, test the alerts
8. Train our L1 team to note all the alerts which come out, and write the action plan for what they need to do in response to each alert
9. Write the training manual for the L1 team with all these alerts and their responses.

5.3 Additional monitoring tools development

10. Execute a software development project to develop all the agents for the parameters written in red above. Develop in sprints:
 - 10.1. Roll out new agents
 - 10.2. Set watermarks for the agents
 - 10.3. Test alerts
 - 10.4. Train the L1 team in the new agents
 - 10.5. Update the training manual with the new agents
11. Hand over full documentation of monitoring tools and parameters to the client.

5.4 L2 support team setup

12. Train the L2 team for L1 support:
 - 12.1. Familiarise them with L1 tasks, tools and processes
 - 12.2. Assign them to work at L1 level in the client team
13. Train the L2 team for scheduled tasks (OS patching, software upgrades, ad hoc queries and reports, etc). Each scheduled task becomes an independent PSvc, supplementary to the main PSvc of managed infra.
 - 13.1. Write the one-page specification for the supplementary PSvc which covers this task
 - 13.2. Assign them to perform one round of the scheduled tasks on client systems
 - 13.3. Update the training manual for this task
 - 13.4. Write the one-page PSvc spec for the task
14. Hand over updated documentation to the client, with all the added tasks and PSvc.

It is key that the L2 team members actually work at L1 level delivering actual service to the client, as part of their training for L2 tasks. This approach is a fundamental orientation method to sensitise higher-level team members to support challenges, tools and processes.

5.5 L3 team setup

15. Train the L3 team:
 - 15.1. Familiarisation training for them in the L1 and L2 tasks, tools and processes (one week)
 - 15.2. Assign them to work at L1 level in the client team (two weeks). The rationale here is the same as for L2 team members working at L1 for two weeks.
 - 15.3. Train them to set up and configure all the software the client is using (e.g. Apache, Tomcat, Postgres, etc) (X weeks)

The customer will begin to see service delivery roughly from W3 or W4, when the L1 team becomes active. The other teams will take more time, but the service may begin, without full tools and processes, by W3 or W4.

7 Outcomes of this setup journey

At the end of this journey of 9-10 weeks, the company will have built the following assets:

- There will be a set of PSvc spec pages, one A4 page per PSvc, describing all the productised services we offer in this area. The top level spec will be for *ServerSage*, and the supplementary ones will be for *ServerSage PatchIt* (OS patch management), *ServerSage Dataplus* (data backups and replication), *ServerSage MoveUp* (controlled software upgrade), *ServerSage HardCore* (technical interventions), *ServerSage ServerPlus* (new servers being set up), *ServerSage FloatBox* (Kubernetes container-based application management), and so on.
- There will be marketing collateral associated with these PSvc, including web pages and technical white papers
- There will be client testimony from our first client, which we can show at other meetings
- There will be training manuals for L1, L2 and L3 for the entire stack of PSvc in the managed infra area.
- There will be a live team whose members can absorb small numbers of new entrants to provide them apprenticeship experience for other client accounts.
- There will be a set of trained trainers who can train new teams for future client contracts.
- We will have the management, control and governance processes in place by which a PM can reliably and efficiently deliver this service to new clients.

* * * * *



Here are five potential smart names for a professional service that manages servers, software, and networks:



1. ServerWise: This name suggests a service that is knowledgeable and experienced in server management and can make wise decisions when it comes to software and network management.
2. NetPro: This name plays on the idea of being a network professional, indicating that the service is highly skilled in managing networks and all the associated software and hardware.
3. SoftServ: This name combines the words "software" and "server" to create a catchy and memorable name that suggests a service that specializes in managing both.
4. SystemSavvy: This name suggests a service that is knowledgeable and experienced in managing complex systems, including servers, software, and networks.
5. TechTactix: This name combines the words "technology" and "tactics" to create a name that suggests a service that is strategic and tactical when it comes to managing servers, software, and networks.



give me five more names



Sure! Here are five more potential names for a professional service that manages servers, software, and networks:



1. NetMinder: This name suggests a service that is vigilant and attentive when it comes to managing networks, always keeping an eye on software and server health.
2. SysGenius: This name implies a high level of expertise and intelligence when it comes to managing systems, including servers, software, and networks.
3. NetForge: This name suggests a service that is skilled at shaping and molding networks to meet the needs of its clients, using software and servers as tools to create a customized network infrastructure.
4. ServerSage: This name implies a service that is wise and experienced when it comes to server management, able to navigate complex software and network issues with ease.
5. NetWorx: This name plays on the idea of a network "workforce," suggesting a service that is capable of managing large and complex networks with many moving parts, including servers and software.