TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Vadim Žigalin 193323IAAB

Implementation of Full Stack Monitoring Service in a Private Telco Cloud Solution on the Example of Elisa Eesti AS

Bachelor's thesis

Supervisors: Siim Vene
MSc
Veiko Koort
Bachelor

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Vadim Žigalin 193323IAAB

Kogupinu seireteenuse juurutamine privaatses Telco Cloud'i lahenduses Elisa Eesti AS'i näitel

Bakalaureusetöö

Juhendajad: Siim Vene
MSc
Veiko Koort
Bakalaureus

Tallinn 2022

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Vadim Žigalin

16.05.2022

Abstract

The aim of this thesis is to implement the first phase of the full stack monitoring service project in the Telco Cloud platform. A unified monitoring environment (suitable for both telecommunications engineers and IT engineers) would remove complexity of troubleshooting and prevent possible service failures.

The first part of the thesis describes the current cloud solution, the problem being solved, and the choice of methodology.

The second part analyzes the current monitoring solution, the need, and requirements for a new monitoring service and technologies for the new service.

The final section focuses on the practical part and conclusion.

This thesis is written in English and is 51 pages long, including 4 chapters, 17 figures and 4 tables.

Annotatsioon

Käesoleva lõputöö eesmärk on viia ellu Telco Cloud'i platvormi kogupinu monitooringu teenuse projekti esimene etapp. Ühtne seirekeskkond (sobiv nii telekommunikatsiooniinseneridele kui ka IT-inseneridele) eemaldaks tõrkeotsingu keerukuse ja hoiaks ära võimalikud teenusetõrked.

Töö esimeses osas kirjeldatakse praegust pilvelahendust, lahendatavat probleemi ja metoodika valikut.

Teises osas analüüsitakse praegust seirelahendust, vajadust ja lähtetingimusi uue seireteenuse järele ning tehnoloogiaid uue teenuse jaoks.

Viimases osas kirjeldatakse praktilist osa ning tehakse järeldus.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 51 leheküljel, 4 peatükki, 17 joonist, 4 tabelit.

List of abbreviations and terms

5G core	A framework of software and protocols for servicing 5G network [1].
CBAM	Nokia CloudBand Application Manager (CBAM) is a single environment for managing applications deployed on a Nokia Telco Cloud [2].
CBIS	Nokia CloudBand Infrastructure Software (CBIS) is a single environment for managing the cloud platform built by Nokia [2].
CEPH	An open-source software-defined block storage.
CloudBand	A modified by Nokia OpenStack environment.
Compute node	A physical node provides computing resources for the Overcloud [3].
CSP	Communications service providers (CSP) is an organization which provides telecommunications services.
Flavor	An OpenStack flavor is a template for virtual machine configuration.
Glance	An OpenStack service for storing images.
Gnocchi	An OpenStack time-series database for storing metrics [4].
Heat	An OpenStack service for launching applications based on templates [5].
IMS	IP Multimedia Subsystem (IMS) is a telecommunications network element which is responsible for SIP protocol routing.
KeyStone	An OpenStack "AAA" service.
NetAct	Nokia software for configuring various elements of a telecommunications network [6].
Nova	An OpenStack service for physical/virtual servers launching.
Nuage Network	A software-defined network solution from Nokia.
Overcloud	Red Hat OpenStack Platform environment [3].
Storage node	A physical node provides storage resources for the Overcloud [3].
Telco Cloud	Cloud computing model specialized in servicing telecommunications services [7].
Undercloud	OpenStack environment for provisioning and managing OpenStack Overcloud [3].
VoLTE	Voice over LTE (VoLTE) is a technology which allows carriers to stream voice services over IP [8].
VSP	Nuage Networks Virtualized Services Platform (VSP) is a software defined network solution offered by Nokia [9].

Table of contents

List of figures

List of tables

1 Introduction

Nowadays cloud solution is one of the most effective ways to allocate resources, manage costs in enterprises and launch new services. Flexibility of configuration and high availability allows to run any service in accordance with business requirements in the cloud [10]. For this reason, it was decided to launch the cloud solution that is specialized for the telecommunications services in Elisa Eesti AS.

1.1 Description of the Telco Cloud and comparison with IT Cloud

Telco Cloud is a concept of the cloud infrastructure which objective is to serve telecommunications services network. The main difference between the Telco Cloud and the IT Cloud is in the criticality of resources and traffic. In case of telecommunications services there those factors like observability, fault tolerance and availability are more important in the Telco Cloud than in the IT cloud. Even a small breakdown can negatively affect the telecommunications services, for example a voice service which is a vital service. Also, the IT cloud can be accessed (depending on application) from external networks, run multiple functions and have different deployment models – private, public or hybrid [11]. However, access to the Telco Cloud must be strictly limited due to the criticality of telecommunications services and data confidentiality. Also, overprovisioning is disabled by default.

The cloud solution used in Elisa Eesti AS is offered by Nokia and deployed in two different data centers providing geo redundancy. Modified Red Hat OpenStack platform is used as a cloud computing solution. Each site is divided into 3 availability zones. There are 20 compute nodes, 6 storage nodes, 5 Nuage VSP (*Virtualized Services Platform*) hypervisors, 3 Overcloud controllers, and the Undercloud controller in each site. The cloud infrastructure configuration as well as installation of additional plugins can be performed by Nokia software – CBIS (*CloudBand Infrastructure Software*) and CBAM (*CloudBand Application Manager*).

1.2 Relevance and features

In this subchapter, the final thesis aim with the basic conditions are set. In additional, features and drawbacks of the Telco Cloud concept are considered.

1.2.1 Telecommunications services design moves closer to IT services design

The main aim of the Telco Cloud solution is to provide Infrastructure as a Service for the telecommunications engineers.

Nowadays, the largest vendors of telecommunications applications are actively promoting the cloud model and offer integration of telecommunications applications into the cloud, guaranteeing a certain level of support [12], [13].

The first reason is that more and more modern telecommunications services (such as 5G core [1]) are using IT protocols instead of telecommunications protocols, becoming more closer to IT.

The second reason is that hardware maintenance and configuration is easier within in the cloud platform. There is no need to build bare-metal platforms anymore for each application. The configuration of new physical servers can be automated by using hardware configuration templates.

1.2.2 High availability

High availability at the physical level is guaranteed by spare physical compute nodes in each datacenter, duplication of the entire storage using RAID 1, self-healing CEPH cluster and locating datacenters in various locations.

The cloud infrastructure makes all the telecommunications services high available at all layers, and it is easier to ensure high availability in the cloud than in the bare metal solution. Both sites are in the same software-defined network and the configuration is centralized, which gives additional reliability and ease of configuration.

High availability at the application layer is designed so that the traffic between zones is distributed evenly, and all application elements communicate with each other. In case of failure of one of the application components, other elements of the same type are ready

to take the traffic of the failed component. Each site in application layer can be loaded up to a maximum of 50% and is always ready to take over another site's traffic in the event of a disaster. This is achieved by long term resource planning.

1.2.3 Scalability of IT infrastructure

The process of scaling any service on the cloud infrastructure is easier than on a bare metal platform. When the telecommunications services were deployed on a bare metal platform it was necessary to order new physical servers and licenses, then the vendor had to reconfigure the entire software solution for the platform. This process was long, expensive, and sometimes even with downtimes.

Now, to scale the telecommunications application, it is enough to provision more resources for the correct tenant and make capacity calculations. Deploying an additional virtual machine using an OpenStack Heat is a simple task for an IT engineer. Further configuration and testing is done the by telecommunications service engineers. Adding new physical resources to the cloud platform is also automated by CBIS. Moreover, migration of virtual machines with the applications can be performed without any downtime [14].

1.3 Problem description and the aim of the thesis

The aim of the thesis is to implement the first phase of the full stack monitoring service development project. During the first phase of the project, it is required to analyze possible technologies and implement the proof of concept. The reactive full stack monitoring service would combine several monitoring services into one and visualize the telecommunications services dependency model. The first phase can be divided into next steps:

1. Analyze the current monitoring solutions.
2. Analyze the basic conditions for the new monitoring service.
3. Plan the high-level design of a new solution via analyzing best practices and possible technologies.

4. Develop a proof of concept of the new service via experiment to be sure that the idea is successful and can be developed in the future.

Generally, the whole project can be divided into next steps:

1. The initial stage of development, confirmation of the proof concept, which is done in the scope of the thesis.

2. Active development stage, implementation of new features.

3. Testing and verification of the result.

4. Additional features analysis.

1.3.1 Reactive and fragmented monitoring system

The main problem with the current monitoring system is the reactivity. This means that all thresholds are static, and alerts are sent only when metric value reached the threshold value. In case of the telecommunications services, the experience of the author and author's team has shown that in most cases, this means that there is already a problem in the service. Also, in the current monitoring system, false positive alarms are often triggered, for example, in the case of running background scripts at night to maintain the health of the cloud platform.

This problem can be solved by creating a proactive monitoring system that could analyze the metrics and analyze the typical behavior of the system to predict possible problems both in the platform and in services launched on the cloud platform.

1.3.2 Complexity of maintenance

The second problem researched in this thesis is the complexity of maintaining the cloud infrastructure. As far as the platform is built on IT technologies, the IT engineers are needed for the platform maintenance. However, the software running on the platform is telecom-specific software and requires engineers of a different specialties to configure and maintain the telecommunications applications. Partially, this problem can be solved by creating a full stack monitoring service which would give a view of all components to both IT engineers and telecommunications engineers. The difficulty of troubleshooting should be significantly reduced through the monitoring service.

1.3.3 Design for large CSPs

The third problem with the Telco Cloud concept is that telecommunications application vendors have fixed types of virtual appliances for the applications, and in most cases, even the smallest appliance is designed for traffic many times more than passes through Elisa Eesti AS network. Thus, Elisa Eesti AS, has no choice, either use systems designed for high traffic, which is expensive, or use outdated systems by own responsibility. All the appliances have fixed requirements for resources. Creation of auto scaling solution for telecommunications application components is not possible due to the complexity of these components, and overprovisioning is not provided by vendors by default.

By final analysis of the effectiveness of the cloud solution, the business team made a hypothesis that the use of resources in the cloud is inefficient. There is no monitoring system that tracks how much resources the virtual machines consume. Because of this it is became necessary to deploy a monitoring solution that will track the amount of resources used by each virtual machine deployed in the cloud.

1.4 Choice of methodology

In this chapter the used methodologies are described.

1.4.1 Comparative analysis

The main methodology is comparative analysis. This analytical method allows to build a logical and understandable organizational scheme for comparison, describe the compared objects with a common ground for comparison, and gives the ability to link different layers of each compared object which gives a deeper understanding of the compared objects [15].

Within the scope of this thesis, the best practices for building monitoring systems are analyzed to acquire the necessary theoretical base, and components of possible monitoring technologies is compared with each other to choose the most suitable solution.

1.4.2 Best practices analyzing and architecture blueprinting

To build a dynamically scalable, automated, proactive, and future-proof solution, the best practices for building monitoring systems with most popular monitoring tools are analyzed [16], [17].

For the detailed understanding of the new service architecture and to avoid possible design errors, a blueprint of the new service is done at a high and at a low level [18].

1.4.3 Proof of concept development via experiment and interview with an expert

Building a proof concept is necessary to make sure that the solution works as planned and is suitable for future development. After the successful deployment of the solution, an interview with other engineers is provided to obtain feedback and verify the solution [16].

2 Solutions analysis

This chapter analyzes the current solution, the basic conditions for the new service, the analysis of the best monitoring practices and possible technologies.

2.1 Basic conditions

The following requirements and restrictions were set for the new service:

- Proactive – in the current monitoring system all the fragments are reactive which means that notifications are sent only in case of real problems. Proactive system means that the notifications should be sent before the possible service failure.

- Both Blackbox and Whitebox techniques – the Blackbox concept is used to check external components, for example, to check the availability of the HTTP service. In the case of the Whitebox concept, the internal components of the service are checked. For example, the number of registered customers in some service. In case of the full stack solution, both technics should be implemented.

- Pull monitoring – according to the best monitoring practices, the most preferable way to collect metrics is pushing data by monitoring agents. But it is possible to implement only monitoring without agents in the provided infrastructure since there is no way to make changes to the monitorable objects, and monitorable objects already have ready-made APIs for data transfer.

- Event, log, and metric-centered – all metrics are stored in a single database. However, despite the convenience of processing data through a single database, additional security methods should be provided to such a database.

- Standard deviation – the problem of reactivity could be partially solved by implementation of dynamic thresholds. The first step to create dynamic thresholds is a basic analysis of the metric through the standard deviation function. Due to

the correct implementation of this function, the monitoring system will give alerts in case of a certain deviation the metric's values.

- Long-term analysis of output data – in addition to the standard deviation function, in the future it will be possible to integrate additional software based on machine learning for in-depth analysis of the metrics and logs, which can provide additional advantages in predicting possible service failures.

- Auto discovery – the new monitoring system should be able to automatically discover new hosts. All the necessary data is already available in the OpenStack database, the new service metrics database will be automatically updated with new hosts.

In addition, the enterprise should avoid the cost of software licenses, which means that the software license should allow to use it for free.

2.2 Relevance and uniqueness of the problem

Since Nokia is one of the largest vendors of telecommunications technologies, including Telco Cloud solutions, researched problem may be relevant for other telecommunications enterprises using the same solution. Also, this solution may be suitable for any other enterprises that use the OpenStack platform, for example, creation a dependency model or centralized monitoring of OpenStack components.

The researched problem and the solution are unique because the next problems are researched in the final thesis:

- Difficult collaboration between telecommunications and IT engineers due to different specializations.

- Lack of proactive monitoring system in Nokia CloudBand platform.

- Lack of a centralized monitoring solution in the Nokia CloudBand solution.

- Final aim is to combine telecommunications and IT services metrics together.

2.3 Current solution

By the moment, the Telco Cloud monitoring system can be divided into three layers:

- Telecommunications services traffic real-time monitoring – provided by Elisa Polystar tools and is used by telecommunications specialists for 24/7 monitoring and engineers for in-depth analysis.
- Telecommunications applications non-real-time analyzing and debugging – managed by Nokia NetAct and is used by telecommunications/radio engineers for troubleshooting and configuration applications/base station.
- Cloud platform physical resources – monitored by Zabbix and used by IT infrastructure engineers to track the cloud platform performance.

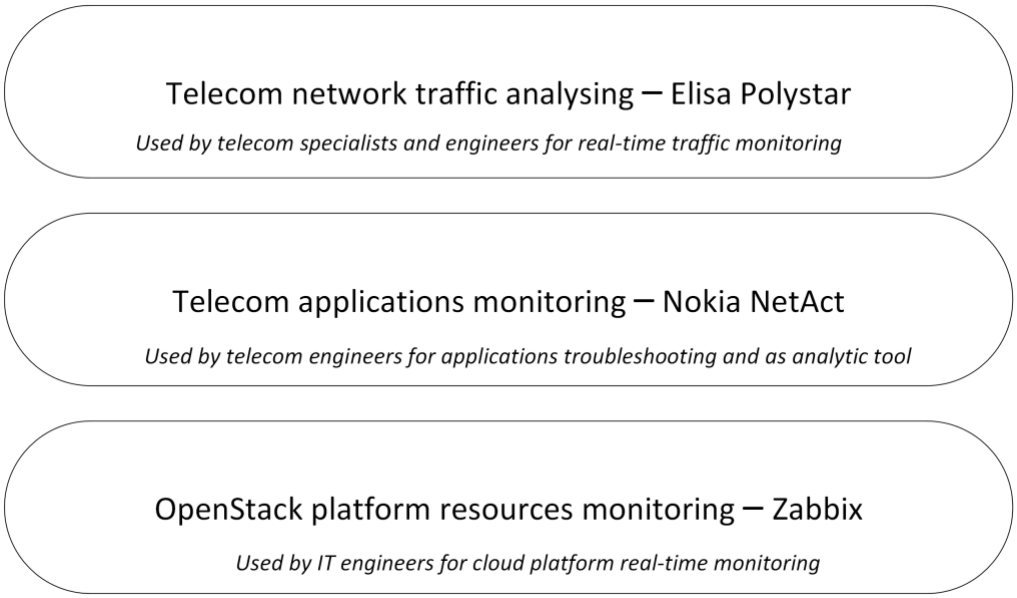


Figure 1. Layers of current monitoring solution.

2.3.1 Telecommunications network level

Elisa Polystar is a set of tools for analyzing, and monitoring CSPs' (*Communications service providers)* network [19]. There are used two Polystar's tools for telecommunications traffic monitoring – KALIX [20] and OSIX [21]. Those tools are used to track performance, analytical information, failures, and even suspicious activity. However, those services show only telecommunications network status, and cannot

automatically detect failures. In case of incorrect metric indicators, the attention of engineers of different classifications is required: IT engineer must troubleshoot the IT platform and telecommunications engineer must troubleshoot the telecommunications service with the application.

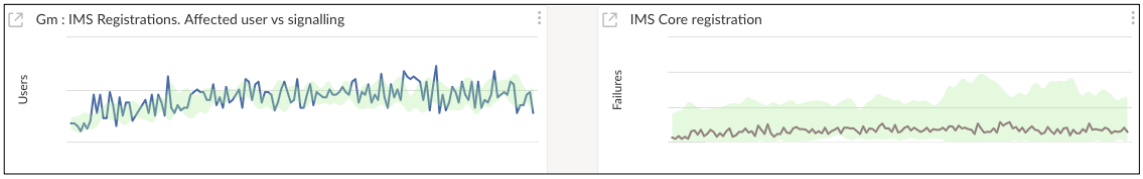


Figure 2. Example of Polystar KALIX tool metrics.

The above example shows which KPIs are used in the tool for IMS core monitoring. It is required to have telecom-specific skills to use the tool.

2.3.2 Telecommunications application level

Monitoring of telecommunications application operations is provided by Nokia NetAct. This is a powerful system for monitoring the status, configuring, and integrating various network elements at all layers. For example, it is possible to configure both the 5G software core (upper layer) and the mobile base station (bottom layer) [6].

However, the main drawback of NetAct is that it works with ready applications, assuming that the application is configured correctly. Thus, if there is an incorrect configuration on the application layer that does not affect the operation of the service, there probably will not be generated any alarms in the NetAct.
It is required to have telecom-specific engineering skills to use the tool, and the provided output data by the tool is incomprehensible even for a standard telecommunications specialist (for example a Network Operations Center specialist).

2.3.3 Cloud platform monitoring

Nokia CloudBand solution provides monitoring of resources and status for several levels and components in the cloud. These are then grouped into 3 main parts: internal components in physical nodes, global monitoring components giving a system overview, and external components communicating with external monitoring systems. All the physical nodes are monitored by Zabbix. Different templates are configured by default for each, depending on functionality of the specific physical host [2].

Zabbix includes an alarm and metering service, and the Zabbix agents provide metering and the status of different processes in each physical node. Elasticsearch provides a data and file collector – beats.
For basic alerting, CBIS can generate SNMP traps via the Zabbix SNMP service. For log exportation, Logstash provides log centralization and exportation management [22], [2].

High	OK	Template App OpenStack Nova: Nova API Metadata Server service is down on {HOST.NAME}	{overcloud-controller-0:proc.num[,,,nova_metadata_w].last(0)}=0
High	OK	Template App OpenStack Nova: Nova API process is not running on {HOST.NAME}	{overcloud-controller-0:proc.num[,,,nova_api_wsgi].last(0)}=0
High	OK	Template App OpenStack Nova: Nova Conductor process is not running on {HOST.NAME}	{overcloud-controller-0:proc.num[,,,nova-conductor].last(0)}=0
High	OK	Template App OpenStack Nova: Nova Console Auth process is not running on {HOST.NAME}	{overcloud-controller-0:proc.num[,,,nova-consoleauth].last(0)}=0
High	OK	Template App OpenStack Nova: Nova NoVnc Proxy process is not running on {HOST.NAME}	{overcloud-controller-0:proc.num[,,,nova-novncproxy].last(0)}=0
High	OK	Template App OpenStack Nova: Nova Scheduler process is not running on {HOST.NAME}	{overcloud-controller-0:proc.num[,,,nova-scheduler].last(0)}=0

Figure 3. Example of Zabbix metrics for Overcloud controllers.

The monitoring solution is designed for IT engineers, it is possible to monitor the resources of the entire cloud. However, in the event of an IT failure, it is very difficult for an IT engineer to understand which telecommunications service is affected by the physical layer failure.

2.3.4 Summary

Each layer of the system has its own monitoring service, but there is no connecting link between them, for example, in the event of a problem at the upper level, it will be difficult to understand whether the problem is somehow related to the lower layer, and vice versa. Also, IT engineers and telecommunications engineers do not have a common part – a single environment that would be clear to everyone, both IT people and telecommunications people. Moreover, it is difficult to predict a failure in the service – Polystar tools analyze only traffic in real time, NetAct monitors only the metrics of the applications and is more used as a configuration solution. Both solutions are telco-specific and do not have the ability to keep track of physical resources. On the other side, there Zabbix, which has a complete look of resources and a very advanced configuration but does not have the ability to associate physical resource consumption with applications.

2.4 Planned solution description and benefits

There is analyzed the need to create a new monitoring service in this paragraph.

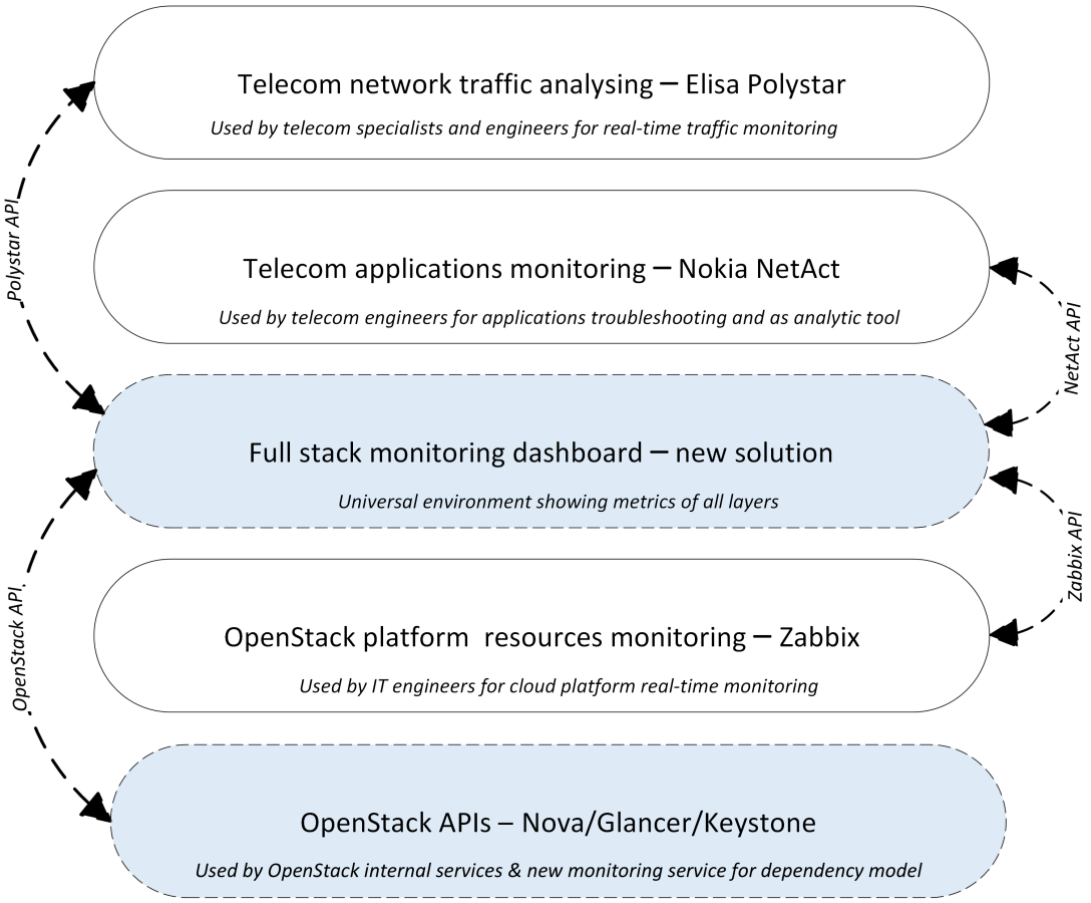


Figure 4. Design of the new monitoring service.

2.4.1 A single dashboard for all engineers

Creating a more versatile dashboard that would be suitable for both telecommunications and IT engineers would help improve communication between engineers, remove complexity of troubleshooting due to differences in the specialization of engineers, and create a more user-friendly UI.

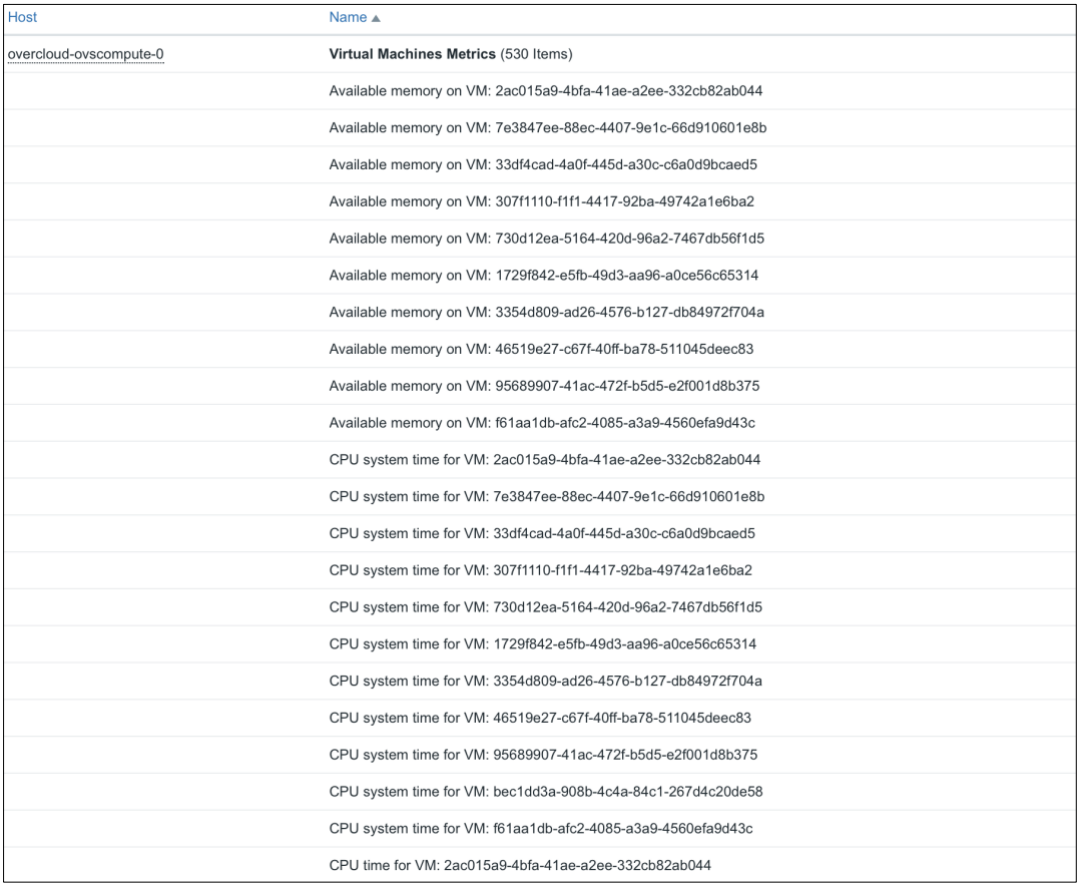
2.4.2 Better virtual machines monitoring

The design of the OpenStack implies that all metrics (including the metrics of virtual machines) are collected through a special single metrics service – Ceilometer and stored in Gnocchi [23], [4]. However, due to performance issues, Nokia has decided to disable

those services in its OpenStack modification and collect virtual machine metrics using Zabbix exporters installed on hypervisors.

This solution has the following disadvantages:

- IDs provided instead of virtual machine names which does not give an idea of what service the machine is responsible for.
- Each hypervisor has a "Virtual Machine Metrics" application attached to it, containing a pool of metrics from all virtual machines on the given hypervisor which provides inconvenience when searching for a specific metric/machine.
- There is no way to make filters by services/machines.



Host	Name ▲
overcloud-ovscompute-0	**Virtual Machines Metrics** (530 Items)
	Available memory on VM: 2ac015a9-4bfa-41ae-a2ee-332cb82ab044
	Available memory on VM: 7e3847ee-88ec-4407-9e1c-66d910601e8b
	Available memory on VM: 33df4cad-4a0f-445d-a30c-c6a0d9bcaed5
	Available memory on VM: 307f1110-f1f1-4417-92ba-49742a1e6ba2
	Available memory on VM: 730d12ea-5164-420d-96a2-7467db56f1d5
	Available memory on VM: 1729f842-e5fb-49d3-aa96-a0ce56c65314
	Available memory on VM: 3354d809-ad26-4576-b127-db84972f704a
	Available memory on VM: 46519e27-c67f-40ff-ba78-511045deec83
	Available memory on VM: 95689907-41ac-472f-b5d5-e2f001d8b375
	Available memory on VM: f61aa1db-afc2-4085-a3a9-4560efa9d43c
	CPU system time for VM: 2ac015a9-4bfa-41ae-a2ee-332cb82ab044
	CPU system time for VM: 7e3847ee-88ec-4407-9e1c-66d910601e8b
	CPU system time for VM: 33df4cad-4a0f-445d-a30c-c6a0d9bcaed5
	CPU system time for VM: 307f1110-f1f1-4417-92ba-49742a1e6ba2
	CPU system time for VM: 730d12ea-5164-420d-96a2-7467db56f1d5
	CPU system time for VM: 1729f842-e5fb-49d3-aa96-a0ce56c65314
	CPU system time for VM: 3354d809-ad26-4576-b127-db84972f704a
	CPU system time for VM: 46519e27-c67f-40ff-ba78-511045deec83
	CPU system time for VM: 95689907-41ac-472f-b5d5-e2f001d8b375
	CPU system time for VM: bec1dd3a-908b-4c4a-84c1-267d4c20de58
	CPU system time for VM: f61aa1db-afc2-4085-a3a9-4560efa9d43c
	CPU time for VM: 2ac015a9-4bfa-41ae-a2ee-332cb82ab044

Figure 5. Screenshot of solution for virtual machines monitoring in Zabbix.

2.4.3 Dependency model and dynamic layout

Creating a dependency model would make it possible to create a logical chain between the cloud platform and the telecommunication services running on the platform. A correct dependency model would make troubleshooting process faster and provide a dynamic virtual machine layout scheme.

Figure 6. An example of the virtual machines layout scheme.

According to the author and other engineers, such a scheme will be useful when the cloud platform becomes more dynamic, and changes become more frequent.

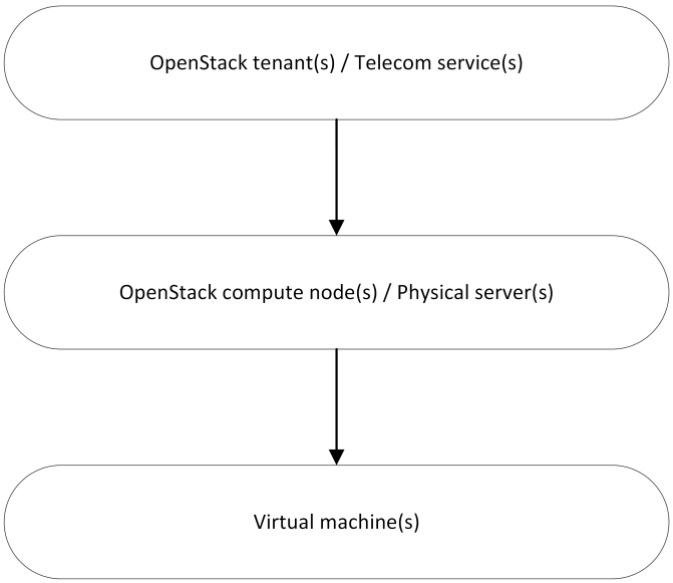


Figure 7. Dependency model design.

Based on provided scheme, initially there is an OpenStack tenant which name shows the telecommunications service, then the physical compute nodes, where are provisioned the service virtual machines, and finally, the virtual machines where are deployed the telecommunications service elements.

2.4.4 Resource optimization in the long term

By analyzing the resource consumption of virtual machines over a long period of time, it will be possible to review each service case by case and perhaps even optimize resource provisioning for specific tenants or virtual machines. The necessary data will be collected by the new monitoring service. In case of a positive result, it will be possible to enable overprovisioning, which will make it possible to manage resources more efficiently.

2.5 Analyze of monitoring best practices

Based on best practices, design of the monitoring service can be divided into several parts. In this subchapter the monitoring best practices analyze is done.

2.5.1 Proactivity and reactivity

Proactivity of the monitoring system means that the monitoring system should be able to detect the possible failure before it appears. This aim can be reached by continuous metric analysis by different software. However, reactivity is also important as a backup plan in

case of incorrect behavior of the proactive system. Static thresholds should be set to critical metric values.

2.5.2 Blackbox and Whitebox monitoring

Blackbox monitoring probes the outside of a service or application. An example of Blackbox monitoring is regularly checking the HTTP service and issuing an alarm in case of any other result than the status "200".

Whitebox monitoring focuses on what's inside the service or application. For example, amount of VoLTE (*Voice over LTE*) registrations. In the case of a working API gateway, it does not mean that the service is working, and in addition to status "200", the number of registered VoLTE subscribers will be regularly checked.

The aim is to make the full stack monitoring service, thus both methodologies will be implemented.

2.5.3 Pull and Push techniques

In most cases, push monitoring systems have several advantages, such as faster hosts auto discovery because agents send all information to a central server, improved security since there is no need to separately listen for connections, and simpler configuration, since there is no need to configure a central server for processing information of various types. Also, the throughput in case of push monitoring is better, since there is no need to download a large amount of information at a time.

However, according to the basic conditions of the problem, the new monitoring service supports only the pull technique, since all the necessary infrastructure has already been configured and it is impossible to change it.

Because the amount of hosts to which pull request are made is limited, all the best practices will be followed:

- Hosts auto discovery – all the OpenStack hosts (both physical and virtual) are stored in single database and available over single REST API request via OpenStack API.

- Security – both endpoints (Zabbix and OpenStack) are protected through credentials verification, one-time tokens, and encrypted requests.

- Configuration – it is enough to configure the collector making requests to the API once, all the necessary information and metrics are dynamically changed in the cloud services.

- Throughput – number of requests are always static, only the amount of transmitted information (JSON) may change in the future.

2.5.4 Standard deviation and dynamic thresholds

In addition to the standard static thresholds, the standard deviation functions will also be applied to each metric, and an alert will be sent in the event of a non-standard value for additional verification. However, any non-standard behavior is not necessarily an indicator of a problem, so with time and experience dynamic thresholds can be adjusted.

2.5.5 Contextual alerts

Any notification from the monitoring system should be comprehensible. For example, it should be considered that the internal processes of the infrastructure or software may cause sometime false alerts that should not be sent. As example, the OpenStack Overcloud controllers can change a master node in the cluster which causes a false alert for a short time. Moreover, every alert should make sense. For example, if the disk is 90% full, it makes no sense to send a single notification that the disk is almost full. It is necessary that the monitoring system would be able to suggest what the root cause might be and send the alert with the suggestion.

2.6 Analyzing of technologies for deployment the proof of concept

To collect, centralize and visualize metrics, four components are required: a collector-software which is available to collect data via HTTP protocol, a time-series database, a metric visualization tool, and a monitoring tool for setting thresholds and alarms sending.

Firstly, a collector is chosen, since for the author the most important is the functionality of the collector and the convenience of configuring, then a database compatible with the collector, and lastly, a monitoring tool compatible with the selected database. The technologies are chosen according to best monitoring practices [16].

2.6.1 Metrics collector

Because the new monitoring service works by pull (agentless) technique, it is required to use a metrics collecting software. The choice was between three technologies:

- Prometheus – author has experience with the service.
- Telegraf – the service is advised by the best practices literature and author has experience with it.
- Collectd – the service is advised by the best practices literature.

The rest of the offered solution in the best practices book were not considered because some of the solutions are available to preform only agent-based monitoring, and others are offered only by commercial license.

Prometheus is the leading open-source solution for monitoring various objects, data storage and visualization [24]. There is a wide range of functionality and a huge number of plugins/exporters to monitor even OpenStack platform [25]. According to the author opinion, this solution is the most suitable, but it is not allowed by Nokia to do any changes in the cloud platform, which means that it is not possible to use Prometheus. As alternative, Prometheus can perform metrics collection via API, but the configuration is too complicated and is not officially supported by Prometheus [26].

Telegraf is a constantly evolving open-source metrics collector that has many integration options and lots of different plugins [27]. It was decided to choose this solution as a metrics collector and not to compare with similar collecting solutions for next reasons:

- Collecting metrics through any API via a configurable HTTP plugin is supported [28].
- Integration with OpenStack and CEPH is supported [29], [30].
- A big number of different plugins, the possibility of increasing the functionality in the future [31].
- Modern solution, constant updates, and the ability to write own plugins, if necessary.

- Local server system logs can be sent to the database via Telegraf [32].
- The author already has experience with configurating Telegraf.

Also, the possibility of using Collectd was advised by best monitoring best practices literature, but author has decided not to use Collectd as a metric collector because according to the author, nowadays it is a legacy solution and will not be expanded in the future.

2.6.2 Time-series database

According to the best practices and author's experience, time-series database is the best solution for storing metric data, since these types of databases are optimized for storing sequence of data points collected over time intervals. The main criteria for comparing databases are the modernity of the solution, the speed of reading data, the disk space usage, and amount of data-processing functions.

Two time-series databases are compared: InfluxDB and QuestDB. Firstly, it is required to make a choice between two different versions of InfluxDB. InfluxDB is a time series database built on its own engine which can be integrated into various data processing tools [33]. The main difference is that the second version of IndluxDB uses a completely different syntax – Flux. Flux is an open-source functional data scripting language and provides the following benefits:

- Support for multiple types of data sources which makes it possible to use this language for queries to other data sources [34].
- Extended functionality which gives more data processing options [35], [36].
- Ability to define custom functions [37].

For the new service monitoring, the "*join()*" and "*pivot()*" functions are especially important because the metric is stored in different tables. For example, metric about OpenStack flavors and Virtual Machines are stored in different tables and a common part is a Flavor ID, for this reason "*join()*" functions is required.

In order the create the same scheme as it is in the "*Figure 6. An example of the virtual machines layout scheme.",* it is required to use "*pivot()*" function due to there is no data

about the virtual machines in the hypervisor table, but table with the virtual machines data contains the information about in which hypervisor the virtual machine is located. An example of the JSON response to the Nova API about virtual machines is provided in the Appendix 2 and proceed by Telegraf response in the Appendix 3.

InfluxQL does not have those functions [38]. Also, InfluxDB v1 support is ended, which means that InfluxDB v1 will not receive any updates in the future [39].

However, despite the advantages, InfluxDB 2 has next disadvantages:

- There is no performance comparison with previous version in the official InfluxDB 2 documentation.
- On the one hand, Flux is a good solution in terms of functionality and will be developed in the future, but on the other hand, the syntax is not SQL-like, and it is required experience with training to use this language.

The author has decided to choose the second version, since this is a newer solution that will receive updates in the future and has more advanced functionality.

The next step is to compare InfluxDB2 and QuestDB. The QuestDB is a time-series database built on the PostgeSQL engine, has many possible integrations. According to the information on the main page, this is the fastest time-series database due to automatic optimization of SQL queries and does not require additional skills therefore this is a relational database [40]. In addition to the performance and ease of use, it is possible to integrate QuestDB via the REST API.

In terms of data processing features, QuestDB has the same number of functions as InfluxDB 2, however, there is no "*pivot()*" function in QuestDB, which makes it difficult to create a dependency model. Based on various tests, QuestDB is about seven times faster than InfluxDB, but the disk space usage of QuestDB is also seven times more than InfluxDB [41], [42].

Despite that QuestDB has a faster read speed it more easy-usable, the author has decided to choose InfluxDB 2 since the necessary functionality is only in the given database. Also, it is possible to deploy InfluxDB 2 as a cluster, but only with the enterprise license [43].

In additional, according to the author experience, the usage of tools from one stack makes the configuration easier and reduces the number of possible bugs.

Table 1. Time-series databases comparison.

Software	Engine	Language	API	Query speed	Disk usage	Cluster	Ease of use	First release
InfluxDB	InfluxDB storage engine	InfluxQL (SQL-like, NoSQL)	InfluxDB API	x	x	Enterprise	Easy	2013
InfluxDB2		Flux (functional data scripting, NoSQL)	InfluxDB v2 API				Specific	2019
QuestDB	PostgreeSQL	SQL (relational)	REST API	~ 7x	~ 7x	Only via K8s	Very easy	2013

2.6.3 Visualization and alerting

There are two tools compared: Grafana and InfluxDB 2. Grafana is a powerful tool for visualizing any data and monitoring metrics, supporting many data sources, visualizations, and custom plugins [44].

InfluxDB 2 uses Chronograf to visualize data and is more narrowly focused to specific metrics and data source. However, InfluxDB supports more advanced metric analysis at the via Kapacitor which is based on a machine-learning [45], [46].

Dynamic variables and dashboards are supported by both solutions [47].

Table 2. Visualization & alerting tools comparison.

Software	Amount of data sources	Amount of visualisations types	Dynamic trasholds alerting	Anomalies detections	Custom plugins
Grafana	20+	20+	Yes	No	Yes
InfluxDB 2	1	10	Only via Kapacitor	Only via Kapacitor	No

As the new monitoring service will process large amount of different metrics, and in addition to InfluxDB 2, the Zabbix API need to be used to track active alerts in real time, the author decided to use Grafana as an alerting and visualization tool. Grafana allows Zabbix integration via RPC API and many data visualization features will be useful for visualizing new telecommunication services metric in the future. Within this thesis, the integration of software for deep analysis of metrics based on machine learning is not considered, and this feature will be implemented later.

3 Implementation of the new monitoring service

This section describes how the new monitoring service is deployed. It covers how the infrastructure is built and services are configurated. The final part covers ideas for the future development.

3.1 Infrastructure high-level design

The primary proof concept has the following logic: Telegraf is used as a metric collector which collects all the metric via OpenStack and Zabbix API. In additional Telegraf listens to local server system logs in format RFC 5423. All the collected logs are sent to InfluxDB 2 via API. Grafana is used as a solution for alarming and processing all the metrics. All the HTTP connections are encrypted.

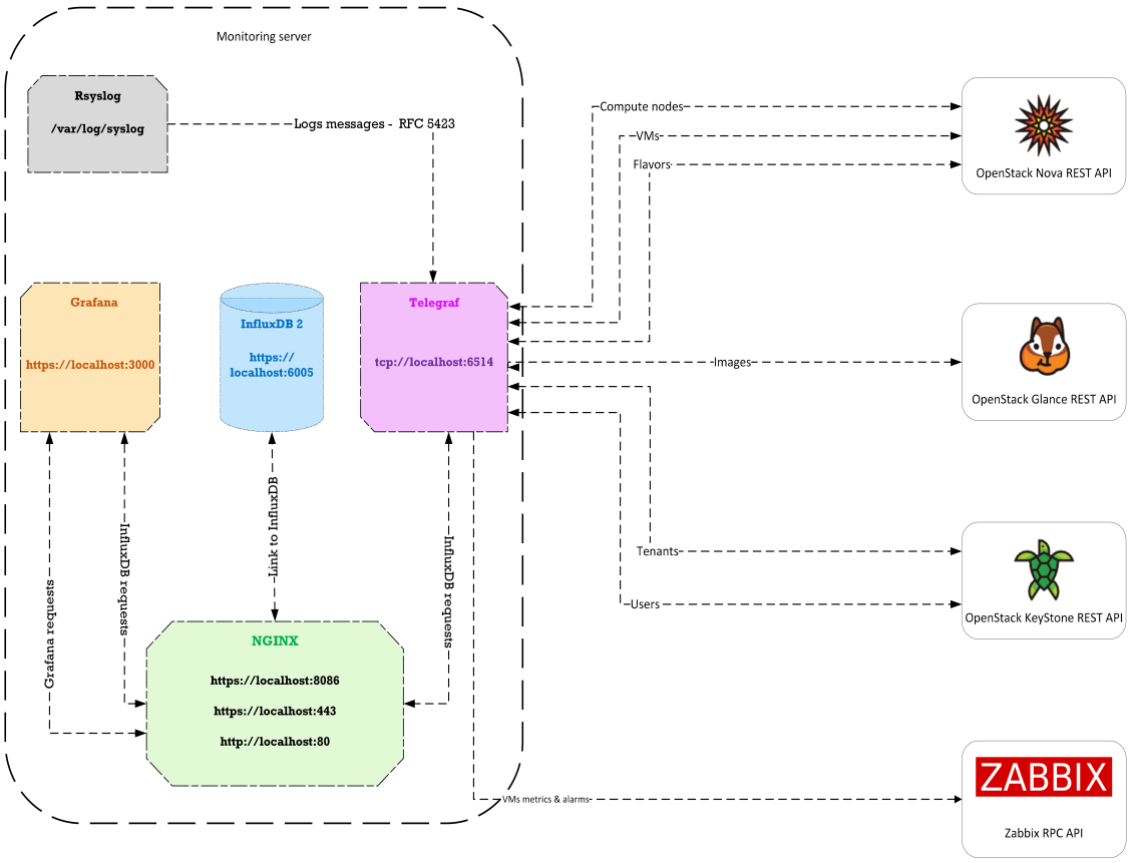


Figure 8. The new monitoring service infrastructure high-level design.

3.2 Telegraf configuration

In this chapter detailed configuration of main used tools are described.

3.2.1 Rsyslog

All the monitoring system logs are collected by Rsyslog and sent to "*tcp://localhost:6514*" via Rsyslog own protocol.

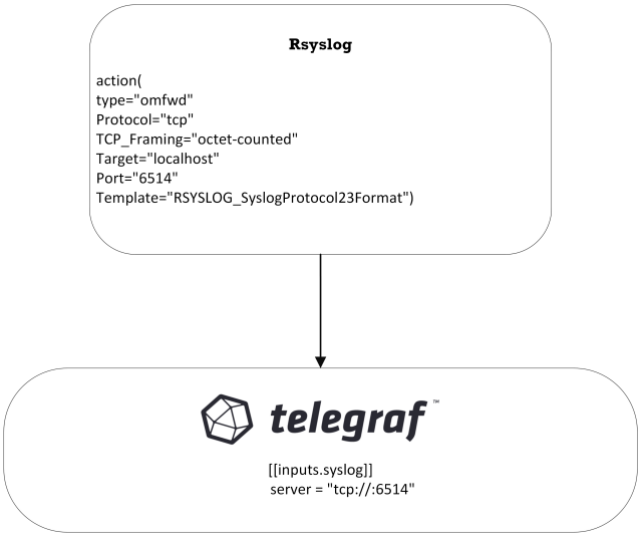


Figure 9. Visualization of the new service infrastructure logging system.

3.2.2 Telegraf authentication and authorization to OpenStack and Zabbix APIs

According to the Telegraf HTTP plugin documentation [28], Telegraf supports basic authentication, cookie authentication, and OAuth2. However, OpenStack uses its own "AAA" service, which cannot be implemented in Telegraf. Keystone service has the following logic:

- Firstly, a POST request with credentials must be sent to the Keystone API endpoint,
- The Keystone service does the user authentication,
- If authentication is successful, the Keystone service makes an authorization for the provided user and returns a JSON with X-Subject-Token,
- The token is valid for next 24 hours and only within the rights of given user.

The problem is that Telegraf does not support an authentication via X-Subject-Token which means that a separate authentication solution should be implemented.

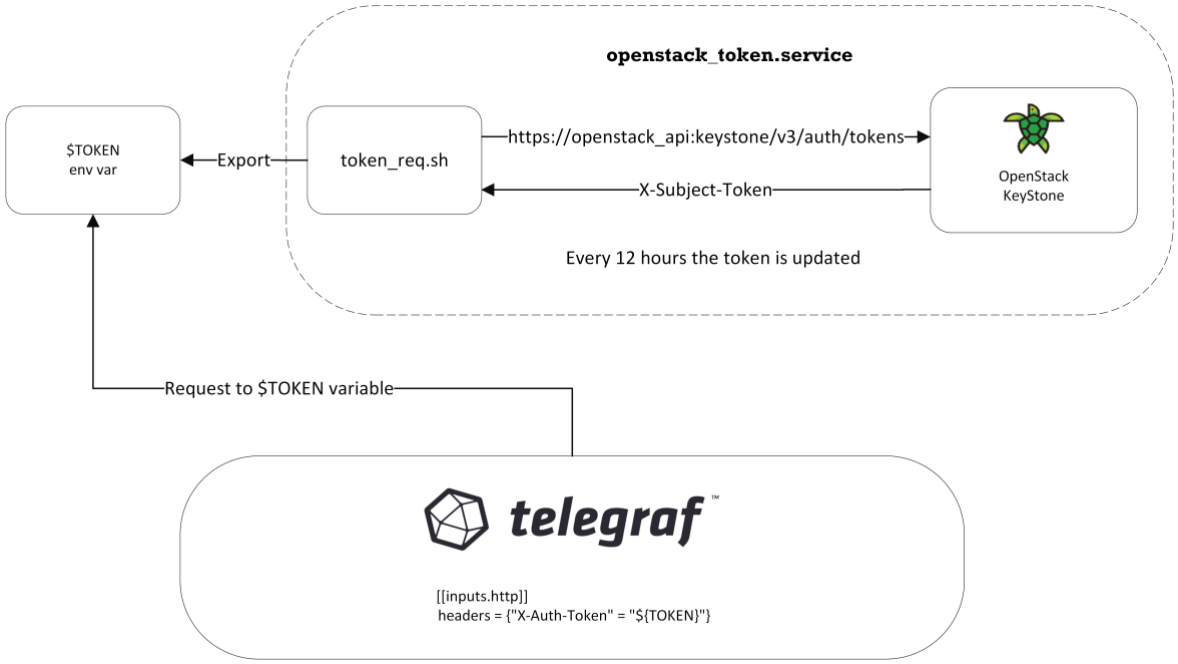


Figure 10. Visualization of the KeyStone token generation process.

Author has decided to implement the token updates via an environment variable. The token is stored in an environment variable and is updated every 12 hours by a special service written by author. The token is used in the Telegraf configuration in the HTTP request header.

By the same logic in built the service for the Zabbix API authentication, because Telegraf does not support authentication to the Zabbix API.

3.2.3 OpenStack API

The challenge of creating a dependency model is that it is required to collect different data from different OpenStack databases. All the data about flavors, physical compute nodes and virtual machines are collected via the Nova service, tenants with users via the Keystone service, and Images via the Glance API.

All the metrics are converted by Telegraf in readable format for the InfluxDB 2 and then sent to the database. Each data typed is stored as a separate measurement (analog of table

in SQL databases). The next table shows the common parts between different measurements which allows to join the measurement.

Table 3. A matrix of common columns in the time-series database tables.

Tenants	Users	VMs	Compute nodes	Flavors	Images	
	Absent	project_id	Absent	Absent	Absent	Tenants
		user_id	Absent	Absent	owner_id	Users
			host	flavor_id	image_id	VMs
				Absent	Absent	Compute nodes
					Absent	Flavors
						Images

The table above shows that the connecting link between any tables is a "virtual machine" measurement. It can be joined with any another measurement. Via the "virtual machine" measurement can be created a link between a tenant (service) and physical server (compute node).

The scheme below shows which how Telegraf collets metric via OpenStack APIs.

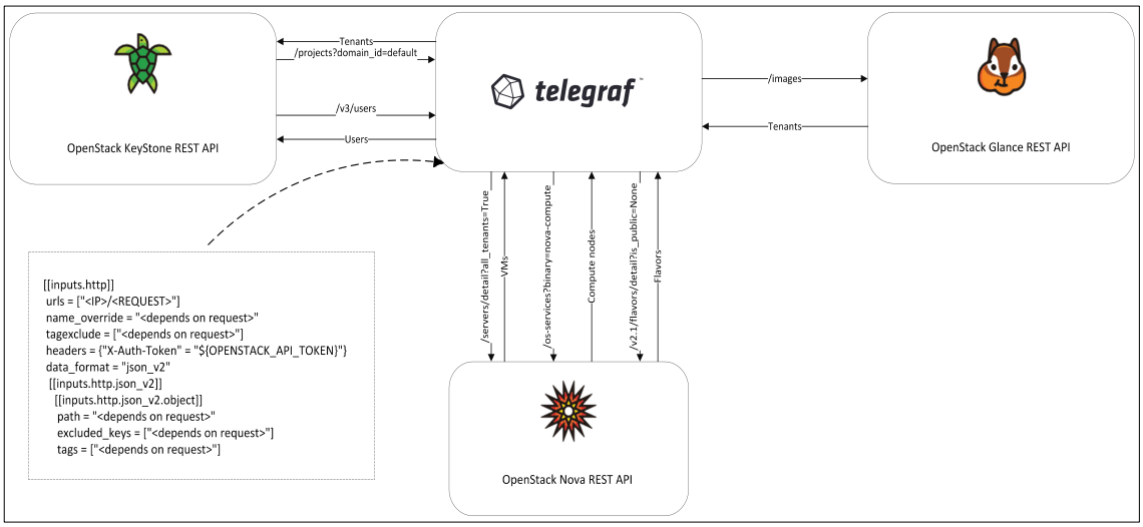


Figure 11. Visualization of collecting metrics via OpenStack APIs.

3.2.4 Zabbix API integration via Telegraf

To solve the mentioned problems with the complexity of monitoring virtual machines via Zabbix, the following solution was found:

- Each virtual machine has 42 different metrics in the Zabbix, each metric is presented as an object which has a unique ID,
- Each item is a collection of time-series metric in the JSON format which can be collected by Telegraf via Zabbix RPC API.

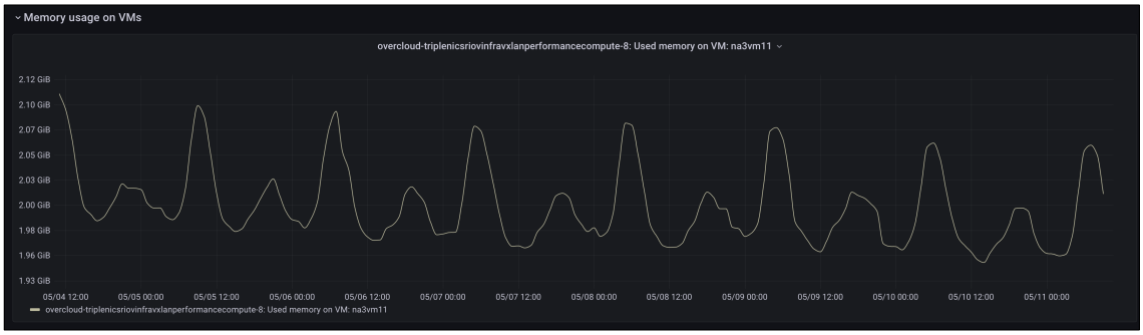


Figure 12. An example of virtual machine metrics visualization.

At the current stage of project, the main aim was to understand is Telegraf available to collect metric via RPC API. The experiment was successful, and the future step would be to automate generation of Telegraf configuration file via Ansible and Jinja2 scripts.

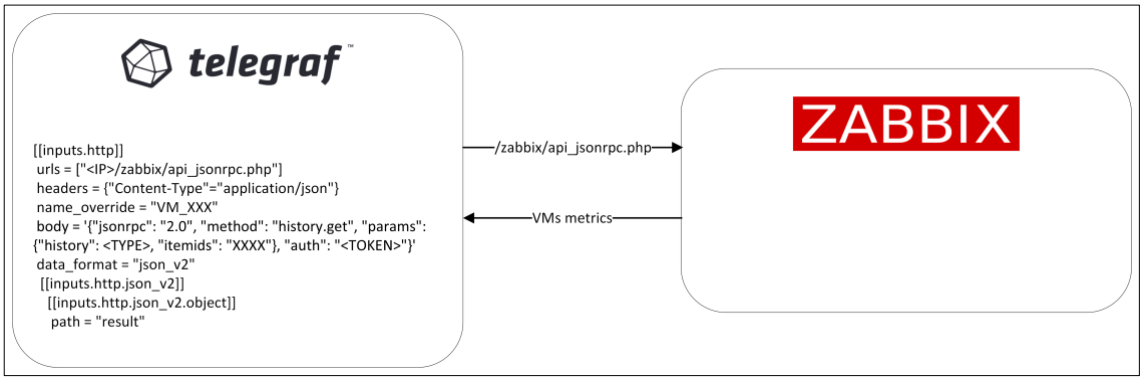


Figure 13. Visualization of metrics collection via Zabbix API.

3.2.5 OpenStack plugin

Despite the fact that the ready-made plugin for OpenStack supports authentication and automatic collection of metrics, it was decided not to use this plugin due to the fact that only data about the flavors and the users are supported from the data necessary to build the dependency model. However, according to the author, in the future this plugin may be useful as it develops, and the plugin can also collect information about networks and hypervisors, which may be useful in the future.

3.2.6 Zabbix alarms via API

The second way to integrate the Grafana with Zabbix is a special plugin [48]. According to the author, using this plugin is much more convenient than using Zabbix, and can be used in the future, for example, to monitor the Zabbix server. At the moment, this plugin is used to centralize Zabbix alerts from different cloud sites.

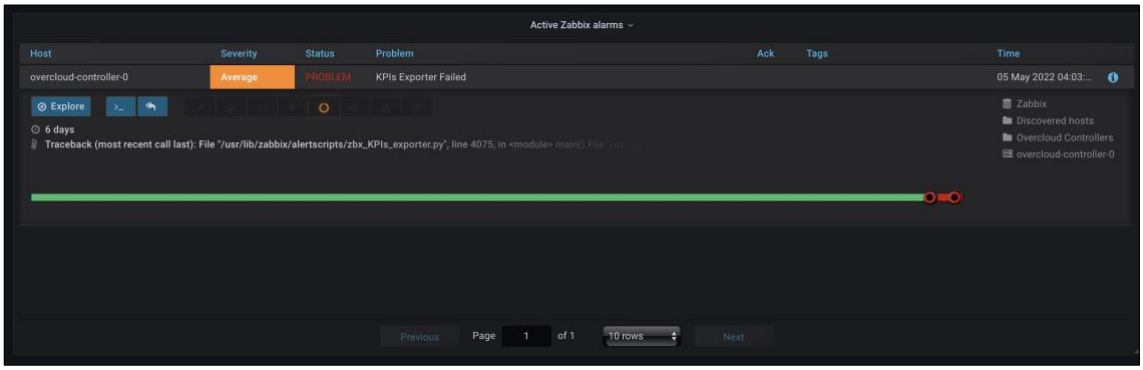


Figure 14. An example of visualization of active cloud platform alarms through the Zabbix API.

The experiment showed that this plugin cannot be used to monitor virtual machines, since the main problem is not solved. It is not possible to filter the metrics and map the names of virtual machines due to single pool.

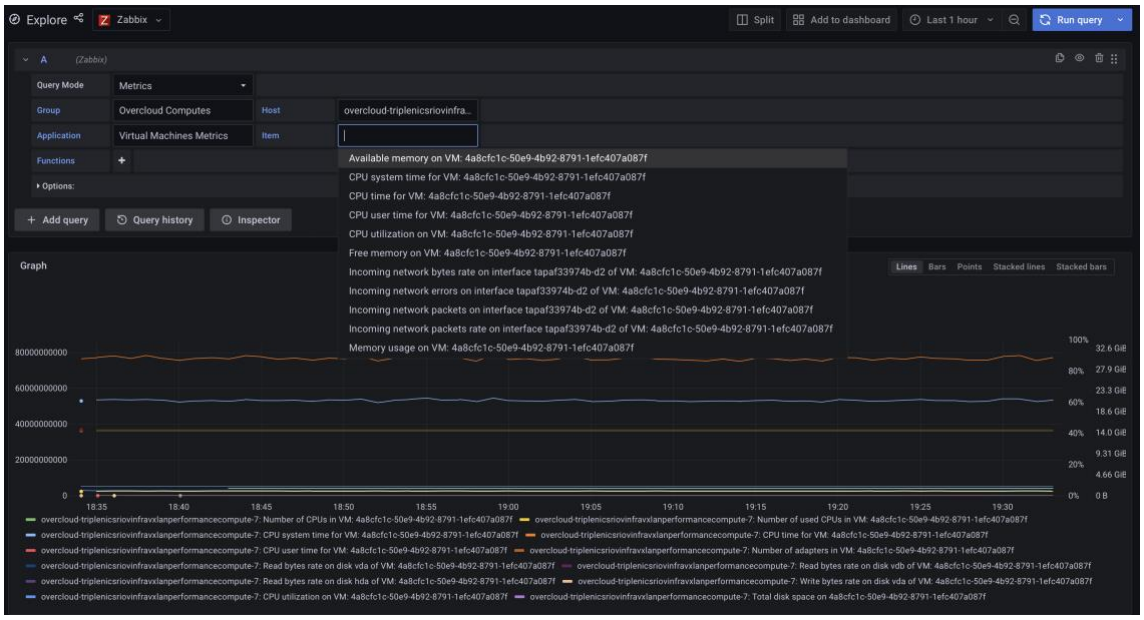


Figure 15. Visualization of the problem collecting virtual machines metrics through the Zabbix API.

3.2.7 Dependency model

Due to the various features of InfluxDB 2, the following dependency model has been able to be composed. In the future, the author plans to add various telecommunications network elements by integrating with Polystar and NetAct.

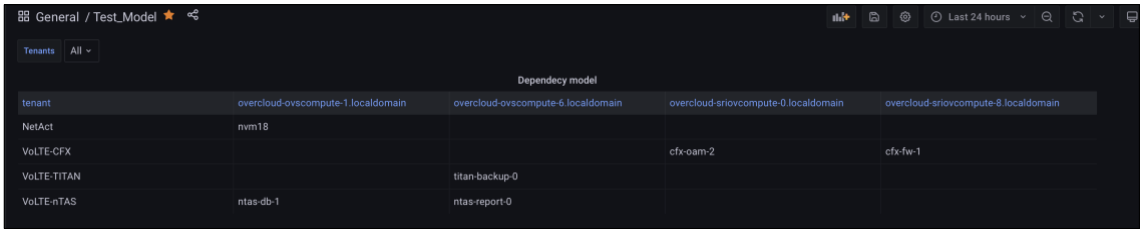


Figure 16. Visualization of the dependency model and the virtual machines layout scheme.

3.3 Infrastructure automatization and monitoring

Automated infrastructure deployment implemented by Ansible. Deployment automation provides the following benefits:

- Configuration backup in case of a disaster.
- Convenient configuration change and infrastructure scaling.

- In the case of implementing automation of mapping the name of virtual machines to their ID, this solution allows you to simply add new machines using a "YAML" file.

- Dynamic configuration files.

The ansible playbook can be found in the Appendix 4.

In order to make sure that monitoring infrastructure is working correctly, the main monitoring server of the enterprise preforms monitoring of following services:

- Grafana service via HTTPS requests.

- InfluxDB 2 via TCP requests.

- CPU load via SNMP v3.

- RAM usage via SNMP v3.

- Disk space usage via SNMP v3.

- Telegraf service via SNMP v3.

- Token generating services via SNMPv3.

3.4 Results validation and benefits to the enterprise

Validation of the result was done by comparing the obtained via new monitoring service metrics with the original ones. For example, the data shown on the dependency model was compared with the original data in OpenStack. The metric data has been compared with the original data in Zabbix. The results obtained showed the identity of the results. Secondary validation was done via an interview with an expert. The author made a presentation of the proof of concept to another team members. The result of the presentation was positive, the implementation of the proof concept was successful. The further development of the new service is approved, also, suggestions were made for the addition of future functionality.

The new centralized dashboard is already helped to detect the memory leak problem at an early stage on some virtual machines after the recent update of the certain service

software. Without a centralized dashboard, the leak would only come out at a late stage since the threshold of memory consumption is set to 90% in the Zabbix configuration by default. Thus, even at the development stage, a small contribution from the project is already made.

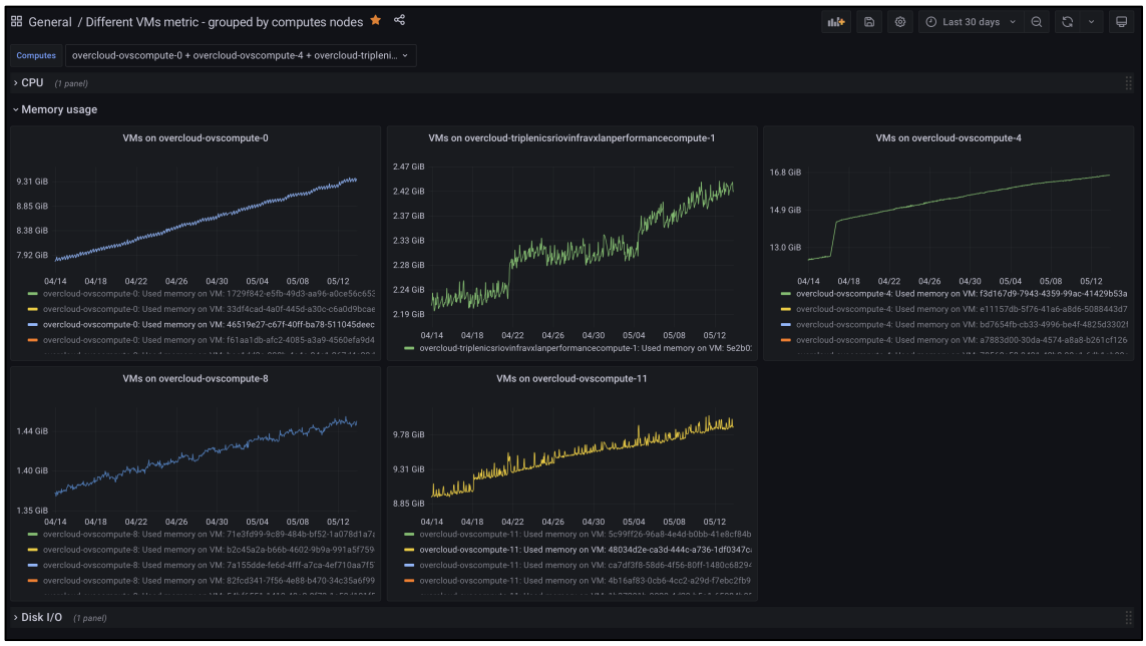


Figure 17. Visualization of detected memory leaks on virtual machines.

In case of Zabbix it would be necessary to check virtual machines one by one (*as shown in the "Figure 5. Screenshot of solution for virtual machines monitoring in Zabbix."*), which would be able to do only an IT engineer. Now, even the telecommunications engineers can track metric of the virtual machines.

3.5 Further development

It is planned about three months to the further development of the project, the feature implementation schedule is next.

Table 4. Further plan for the development of the new monitoring service.

Task	Complexity	Time
Add automat mapping of virtual machine names to IDs and adding metrics for each machine as a separate measurement via Ansible and Jinja2	Medium	3 days
Add physical server metrics	Easy	2 days
Polystar & NetAct integration	Medium	1-2 weeks
Add telecommunications services suitable metrics	Hard	2 weeks
Develop new KPIs (if necessary)	Hard	2-3 weeks
Logstash integration	Medium	1 week
Dynamic thresholds development via standard deviation	Hard	2-3 weeks
Result validation	Medium	2 weeks
Machine-learning software integration (if necessary)	Hard	

The primary aim is to develop final automated solution for virtual machines monitoring. To the author's opinion, this task should not be a challengeable because the logic of integration virtual machines metric in already found and now should be automated. Second task is effortless due to the physical server's metrics can be parsed directly via Zabbix API. However, all the next tasks are challengeable and may take some time for implementation. According to the author, all project implementations will take about 11-14 weeks.

After the project realization, integration of machine-learning and monitoring infrastructure to microservices will be considered.

4 Summary

The aim of the thesis is achieved. The analysis of the best monitoring practices and possible technologies was done. The proof of concept also was deployed which corresponds to the basic conditions. The first phase of the project was successfully completed, and the hypothesis about the possibility of creating a full stack monitoring service via different APIs in the Telco Cloud by Nokia is proven. Also, the contribution from the project has already been partially implemented. The memory leak was detected at an early stage, which prevented a possible failure in the telecommunications service.

The author received a good theoretical base regarding monitoring practices, gained practical experience in choosing technologies, working with API services, and visualizing metrics. The author also learned how to perform a correct analyze.

It is positive that in the future the implementation of this project is possible, and the final of this project will bring a significant contribution to the development of Telco Cloud solution for Elisa Eesti AS.

References

[1] Nokia Corp., "Nokia 5G Core (5GC)," [Online]. Available: https://www.nokia.com/networks/portfolio/5g-core/. [Accessed 15 May 2022].

[2] Nokia Corp., "CBIS 20 Operation Manual - Monitoring," [Online]. Available: https://doc.networks.nokia.com/product/CloudBand_Infrastructure/833-003525.00/release/CBIS_20/833-063841.00/document/CBIS_Operation_Manual/cbis20sp2C5June2021--cbis_operation_manual/topic/sky-uJ2Cv. [Accessed 15 May 2022].

[3] Red Hat Customer Portal, "Red Hat OpenStack Platform — Chapter 1. Introduction," [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/9/html/director_installation_and_usage/chap-introduction. [Accessed 15 May 2022].

[4] OpenStack Wiki, "Gnocchi," [Online]. Available: https://wiki.openstack.org/wiki/Gnocchi. [Accessed 15 May 2022].

[5] OpenStack Wiki, "OpenStack Orchestration," [Online]. Available: https://wiki.openstack.org/wiki/Heat. [Accessed 15 May 2022].

[6] Nokia Corp., "NetAct — Embrace the evolution to 5G," [Online]. Available: https://www.nokia.com/networks/products/netact/. [Accessed 15 May 2022].

[7] Red Hat, "What is telco cloud?," [Online]. Available: https://www.redhat.com/en/topics/cloud-computing/what-is-telco-cloud. [Accessed 15 May 2022].

[8] Nokia Corp., "Voice over LTE (VoLTE) and Voice over Wifi (VoWi-Fi) core," [Online]. Available: https://www.nokia.com/networks/solutions/voice-over-lte-volte-and-voice-over-wifi-vowi-fi-core/. [Accessed 15 May 2022].

[9] Nokia Corp., "Nuage Networks Virtualized Services Platform," [Online]. Available: https://www.nuagenetworks.net/platform/virtualized-services-platform/. [Accessed 15 May 2022].

[10] Nokia Corp., "Dynamic Enterprise Services - Cloud services," [Online]. Available: https://www.nokia.com/networks/dynamic-enterprise-services/cloud-services/. [Accessed 15 May 2022].

[11] Amazon Web Services Incorporation, "Cloud Computing Deployment Models," [Online]. Available: https://aws.amazon.com/types-of-cloud-computing/. [Accessed 15 May 2022].

[12] Telefonaktiebolaget LM Ericsson, "Cloud Infrastructure for 5G," [Online]. Available: https://www.ericsson.com/en/cloud-infrastructure. [Accessed 15 May 2022].

[13] Nokia Corp., "Going cloud native," [Online]. Available: https://www.nokia.com/networks/cloud-native/. [Accessed 15 May 2022].

[14] OpenStack Documentation, "Live-migrate instances," [Online]. Available: https://docs.openstack.org/nova/pike/admin/live-migration-usage.html. [Accessed 15 May 2022].

[15] K. Walk, "How to Write a Comparative Analysis," Writing Center at Harvard University, 1992. [Online]. Available: https://writingcenter.fas.harvard.edu/pages/how-write-comparative-analysis. [Accessed 15 May 2022].

[16] J. Turnbull, The Art of Monitoring, Brooklyn: Turnbull Press, 2016.

[17] M. Julian, Practical Monitoring, Sebastopol: O'Reilly Media, 2017.

[18] G. Booch, J. Rumbaugh and I. Jacobson, "Chapter 1. Why We Model," in *Unified Modeling Language User Guide, The, Second Edition*, Boston, Addison-Wesley Professional, 2005.

[19] Elisa Polystar HQ, "Self-driving network products in cloud," [Online]. Available: https://www.elisapolystar.com/products/. [Accessed 15 May 2022].

[20] Elisa Polystar HQ, "KALIX – Agile data insights for network operators," [Online]. Available: https://www.elisapolystar.com/kalix/. [Accessed 15 May 2022].

[21] Elisa Polystar HQ, "OSIX – End-to-End network monitoring with deep protocol drill down analytics," [Online]. Available: https://www.elisapolystar.com/osix/. [Accessed 15 May 2022].

[22] Nokia EDU, "CBIS Monitoring Architecture Overview," in *CBIS Description - Student Guide*, 2018.

[23] OpenStack Documentation, "Welcome to Ceilometer's documentation!," [Online]. Available: https://docs.openstack.org/ceilometer/latest/. [Accessed 15 May 2022].

[24] Prometheus Authors, "From metrics to insight. Power your metrics and alerting with the leading open-source monitoring solution.," [Online]. Available: https://prometheus.io. [Accessed 15 May 2022].

[25] Prometheus Authors, "Exporters and integrations," [Online]. Available: https://prometheus.io/docs/instrumenting/exporters/. [Accessed 15 May 2022].

[26] H. Davtyan, "Transforming remote JSON into Prometheus metrics," Level Up Coding, 25 February 2022. [Online]. Available: https://levelup.gitconnected.com/transforming-remote-json-into-prometheus-metrics-334d772df38a. [Accessed 14 May 2022].

[27] Influxdata Inc., "Telegraf is the open source server agent to help you collect metrics from your stacks, sensors, and systems.," [Online]. Available: https://www.influxdata.com/time-series-platform/telegraf/. [Accessed 15 May 2022].

[28] Influxdata, "HTTP Input Plugin," [Online]. Available: https://github.com/influxdata/telegraf/tree/release-1.22/plugins/inputs/http. [Accessed 15 May 2022].

[29] Influxdata, "OpenStack Input Plugin," [Online]. Available: https://github.com/influxdata/telegraf/tree/release-1.22/plugins/inputs/openstack. [Accessed 15 May 2022].

[30] Influxdata, "Ceph Storage Input Plugin," [Online]. Available: https://github.com/influxdata/telegraf/blob/release-1.22/plugins/inputs/ceph/README.md. [Accessed 15 May 2022].

[31] Influxdata Documentation, "Plugin directory - Input plugins," [Online]. Available: https://docs.influxdata.com/telegraf/v1.22/plugins/#input-plugins. [Accessed 15 May 2022].

[32] Influxdata, "Telegraf - Syslog Input Plugin," [Online]. Available: https://github.com/influxdata/telegraf/blob/release-1.22/plugins/inputs/syslog/README.md. [Accessed 15 May 2022].

[33] Influxdata Documentation, "InfluxDB tools and integrations," [Online]. Available: https://docs.influxdata.com/influxdb/v2.1/tools/. [Accessed 15 May 2022].

[34] Influxdata Documentation, "Flux documentation," [Online]. Available: https://docs.influxdata.com/flux/v0.x/. [Accessed 15 May 2022].

[35] Influxdata Documentation, "Complete list of Flux functions," [Online]. Available: https://docs.influxdata.com/flux/v0.x/stdlib/all-functions/. [Accessed 15 May 2022].

[36] Influxdata Documentation, "Flux vs InfluxQL," [Online]. Available: https://docs.influxdata.com/influxdb/v1.8/flux/flux-vs-influxql/. [Accessed 15 May 2022].

[37] Influxdata Documentation, "Define custom functions," [Online]. Available: https://docs.influxdata.com/flux/v0.x/define-functions/. [Accessed 15 May 2022].

[38] Influxdata Documentation, "InfluxQL functions," [Online]. Available: https://docs.influxdata.com/influxdb/v1.8/query_language/functions/. [Accessed 15 May 2022].

[39] Influxdata Inc., "InfluxDB OSS and Enterprise Roadmap Update from InfluxDays EMEA," [Online]. Available: https://www.influxdata.com/blog/influxdb-oss-and-enterprise-roadmap-update-from-influxdays-emea/. [Accessed 15 May 2022].

[40] QuestDB, "Fast SQL for time series," [Online]. Available: https://questdb.io. [Accessed 15 May 2022].

[41] F. Pardi, "A Timeseries Case Study: InfluxDB VS PostgreSQL to store data," Portavita Devblog, 31 July 2018. [Online]. Available: https://portavita.github.io/2018-07-31-blog_influxdb_vs_postgresql. [Accessed 15 May 2022].

[42] QuestDB, "Comparing InfluxDB and QuestDB databases," [Online]. Available: https://questdb.io/tutorial/2021/11/29/questdb-versus-influxdb/. [Accessed 15 May 2022].

[43] Influxdata Inc., "InfluxDB Editions," [Online]. Available: https://www.influxdata.com/products/editions/. [Accessed 15 May 2022].

[44] Grafana Labs, "Dashboard anything. Observe everything.," [Online]. Available: https://grafana.com/grafana/. [Accessed 15 May 2022].

[45] Influxdata Inc., "Kapacitor," [Online]. Available: https://www.influxdata.com/time-series-platform/kapacitor/. [Accessed 15 May 2022].

[46] Influxdata Documentation, "Custom anomaly detection using Kapacitor," [Online]. Available: https://docs.influxdata.com/kapacitor/v1.6/guides/anomaly_detection/#detecting-anomalies. [Accessed 15 May 2022].

[47] MetricFire Corp., "Grafana vs Chronograf - The Dashboards," 17 June 2020. [Online]. Available: https://www.metricfire.com/blog/grafana-vs-chronograf-the-dashboards/. [Accessed 15 May 2022].

[48] A. Zobnin, "Zabbix," Grafana Labs, [Online]. Available: https://grafana.com/grafana/plugins/alexanderzobnin-zabbix-app/?tab=changelog. [Accessed 15 May 2022].

Appendix 1 – Non-exclusive license for reproduction and publication of a graduation thesis[1]

I Vadim Žigalin

1. Grant Tallinn University of Technology free license (non-exclusive license) for my thesis “Implementation of Full Stack Monitoring Service in a Private Telco Cloud Solution on the Example of Elisa Eesti AS”, supervised by Siim Vene and Veiko Koort:
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive license.
3. I confirm that granting the non-exclusive license does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

16.05.2022

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 – An example of JSON reply from Nova API with virtual machines data [1]


```
{
  "servers": [
    {
      "OS-EXT-STS:task_state": null,
      "addresses": {
        "VNF-IntNet0": [
          {
            "OS-EXT-IPS-MAC:mac_addr": "XX:XX:XX:XX:XX:XX",
            "version": 4,
            "addr": "XXX.XXX.XXX.XXX",
            "OS-EXT-IPS:type": "fixed"
          }
        ],
        "SERVICE-VNF-OAM-EE-SITE": [
          {
            "OS-EXT-IPS-MAC:mac_addr": "XX:XX:XX:XX:XX:XX",
            "version": 4,
            "addr": "XXX.XXX.XXX.XXX",
            "OS-EXT-IPS:type": "fixed"
          }
        ]
      },
      "links": [
        {
          "href": "https://XXX.XXX.XXX.XXX:YYYYY/v2.1/servers/d2e5825e",
          "rel": "self"
        },
        {
          "href": "https:// XXX.XXX.XXX.XXX:YYYYY/servers/d2e5825e",
          "rel": "bookmark"
        }
      ],
      "image": {
        "id": "b77f6035-e8a4-49f6-8afa-695ebd05a057",
        "links": [
          {
            "href": "https://XXX.XXX.XXX.XXX:YYYYY/images/b77f6035",
            "rel": "bookmark"
          }
        ]
      },
      "OS-EXT-STS:vm_state": "active",
      "OS-EXT-SRV-ATTR:instance_name": "instance-000010a6",
      "OS-SRV-USG:launched_at": "2022-04-21T14:07:24.000000",
      "flavor": {
        "id": "502b9487-c0e9-4d4d-827f-ac06191698c4",
        "links": [
          {
            "href": "https://XXX.XXX.XXX.XXX:YYYYY/flavors/502b9487",
            "rel": "bookmark"
          }
        ]
      },
      "id": "d2e5825e-3e81-41d4-b257-36303f2aa1b4",
      "security_groups": [
        {
          "name": "VNF-OpenSecGroup"
        },
```


1 The data provided only as an example, confidential information has been removed.

```
        {
          "name": "VNF-OpenSecGroup"
        }
      ],
      "user_id": "b0eee7a1dd7248bca8b28dc7b4f87927",
      "OS-DCF:diskConfig": "MANUAL",
      "accessIPv4": "",
      "accessIPv6": "",
      "progress": 0,
      "OS-EXT-STS:power state": 1,
      "OS-EXT-AZ:availability_zone": "zoneX",
      "config_drive": "True",
      "status": "ACTIVE",
      "updated": "2022-04-21T14:07:25Z",
      "hostId": "569f10a37bcd281a331",
      "OS-EXT-SRV-ATTR:host": "compute-X.localdomain",
      "OS-SRV-USG:terminated_at": null,
      "key_name": null,
      "OS-EXT-SRV-ATTR:hypervisor_hostname": "compute-X.localdomain",
      "name": "VNF-oam-a",
      "created": "2022-04-21T14:07:02Z",
      "tenant_id": "73583f7f",
      "os-extended-volumes:volumes_attached": [
        {
          "id": "026631d0"
        }
      ]
    }
  ]
}
```

Appendix 3 – An example of proceeded by Telegraf JSON reply from Nova API with virtual machines data [1]

```
name=SERVICE-oam03,
OS-DCF:diskConfig="MANUAL",
OS-EXT-AZ:availability_zone="zoneX",
OS-EXT-SRV-ATTR:host="compute-X.localdomain",
OS-EXT-SRV-ATTR:hypervisor_hostname="compute-X.localdomain",
OS-EXT-SRV-ATTR:instance name="instance-0000043e",
OS-EXT-STS:power state=1,
OS-EXT-STS:vm_state="active",
OS-SRV-USG:launched_at="2021-02-09T10:41:07.000000",
accessIPv4="",
accessIPv6="",
config drive="True"
created="2021-02-09T10:39:31Z",
flavor_id="06df2855-1b71-4ed2-addb-8e8d73c85ccd",
hostId="3336831e58f720065fa235d4476b0b056f2e932800b9a7890ceca19b",
d="1729f842-e5fb-49d3-aa96-a0ce56c65314",
image_id="db7be811-0b80-4072-a93c-d9af0dddcace",
key name="SERVICE-537ff7280a5b4b42be51792e815aa434",
metadata flavor="VNF MANAGEMENT 4 32 42",
metadata_image="oam_2100933"
```

1 The data provided only as an example, confidential information has been removed.

Appendix 4 – Ansible playbook for the infrastructure

```
---
- name: General configuration
  hosts: main
  become: yes
  roles:
    - general_configuration
  tags:
    - general

- name: NGINX
  hosts: main
  become: yes
  roles:
    - nginx
  tags:
    - nginx

- name: InfluxDB
  hosts: main
  become: yes
  roles:
    - influxdb
  tags:
    - influxdb

- name: Grafana
  hosts: main
  become: yes
  roles:
    - grafana
  tags:
    - grafana

- name: Telegraf
  hosts: main
  become: yes
  roles:
    - telegraf
  tags:
    - telegraf

- name: Rsyslog
  hosts: main
  become: ye
  roles:
    - rsyslog
  tags:
    - rsyslog

- name: Snmpd
  hosts: main
  become: yes
  roles:
    - snmpd
  tags:
    - snmpd
```