D6.1 - Infrastructure Operation Report

Editor:	Michel Corriou, B-COM	
Deliverable nature:	Report (R)	
Dissemination level:	Public (PU)	
Date: planned | actual	21 December 2018	20 December 2018
Version | No. of pages	1.0	34
Keywords:	Ticketing, Monitoring, KPI	

Abstract

This document contains the report on the operational status of the infrastructure, including statistics on usage, reliability and performance as well as the tooling that is used to operate and monitor the infrastructure. The work of this report is contributed by tasks: Task 6.1 - Experimental Facilities Operations; Task 6.2 - Infrastructure maintenance and Issue resolution.

Disclaimer

This document contains material, which is the copyright of certain 5GINFIRE consortium parties, and may not be reproduced or copied without permission.

All 5GINFIRE consortium parties have agreed to full publication of this document.

Neither the 5GINFIRE consortium as a whole, nor a certain part of the 5GINFIRE consortium, warrant that the information contained in this document is capable of use, nor that use of the information is free from risk, accepting no liability for loss or damage suffered by any person using this information.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732497. This publication reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

Impressum

Full project title: Evolving FIRE into a 5G-Oriented Experimental Playground for Vertical Industries

Short project title: 5GINFIRE

Number and title of work-package: WP6 Infrastructure Operation

Number and title of task:

- Task 6.1 - Experimental Facilities Operations
- Task 6.2 - Infrastructure maintenance and Issue resolution

Document title: D6.1 - Infrastructure Operation Report

Editor: Michel Corriou, B-COM

Work-package leader: Michel Corriou, B-COM

Copyright notice

2018 B-COM and members of the 5GINFIRE consortium

Executive summary

This deliverable provides the report on the status of the infrastructure including statistics on usage, reliability and performance. It describes also the tools and the processes used to monitor and produce the KPIs.

It is the outcome of the activities driven inside the Work Package, namely:

- Task 6.1 - **Experimental Facilities Operations**: addresses day-to-day infrastructure monitoring and maintenance operations.
- Task 6.2 - **Infrastructure maintenance and Issue resolution**: provides the means to interact between the different 5GINFIRE stakeholders through the helpdesk tool.

These activities rely on tools provided by other 5GINFIRE Work Packages like the 5GINFIRE portal, the VNF Continuous Integration pipeline, the OSM monitoring framework and the infrastructure interconnection performance framework. For all, a high level description will be provided in this deliverable together with processes to collect the required KPIs.

The report also provides some intermediate KPIs at M24 about 5GINFIRE infrastructure. The analysis of those KPIs demonstrates:

- A first set of tools including ticketing tool based on Bugzilla has been deployed and are used by the 5GINFIRE community. With infrastructure and connectivity status monitoring, this was the priority in order to enable experimentation hosting.
- The process to manage the support to experimentation is applied and the time-to-respond has been improved between Q3 and Q4 2018.
- The open tickets are followed during regular WP6 meetings.
- Many core components like 5GINFIRE Portal, OSM and some testbeds like 5TONIC, Bristol, IT-AV and eHealth5G have excellent availability scores above 80% of availability.

The availability ratio of an experimentation infrastructure shall not be compared with a production environment which targets the five nine (99,999%). The availability ratio of an experimentation infrastructure shall be interpreted taking into account that such an infrastructure is operated 5 days a week and only during open office hours and that the experiment execution is agreed with the experimenters.

The report also underlines the targets for enhancements in the next project period:

- Improve the 5GINFIRE infrastructure monitoring with usage and performance KPIs
- Automate data collection (2018 Q3 and Q4 statistics are calculated after data post-processing)
- Add logs and events data storage

For those enhancements, tools have already been identified and prototyped.

List of authors

Company	**Author**
B-COM	Michel Corriou, Sergio Morant
UNIVERSITY OF PATRAS	Christos Tranoris
INSTITUTO DE TELECOMUNICAÇÕES	Diogo Gomes, Sandra Moreira
UNIVERSIDAD CARLOS III DE MADRID	Luis Félix González, Iván Vidal
TELEFONICA I+D	Juan Rodriguez Martinez

Table of Contents

List of figures and tables

Abbreviations

5G	5th Generation
AGPL	GNU Affero General Public License
API	Application Programming Interface
BSS	Business Support System
CI/CD	Continuous Integration/Continuous Deployment
ETSI	European Telecommunications Standards Institute
EVI	Experimental Vertical Instance
GPL	GNU General Public License
GUI	Graphical User Interface
HCS	Health Check Service
IPMI	Intelligent Platform Management Interface
MANO	Management and orchestration
NBI	Northbound Interface
NFV	Network Function Virtualization
NFVI	NFV Infrastructure
NFVO	NFV Orchestrator
NSO	Network Service Orchestrator
OS	Operating System
OSM	Open Source Mano
OSS	Operations Support System
REST	Representational state transfer
RO	Resource orchestrator
SBI	Southbound interfaces
SDN	Software Defined Network
SNMP	Simple Network Management Protocol
SFC	Service Function Chaining
TOSCA	Topology and Orchestration Specification for Cloud Applications
vCPU	virtual Central Processing Unit
VIM	Virtual Infrastructure Management
VNF	Virtual network function
VNFM	VNF Manager
YAML	YAML Ain't Markup Language

1 Introduction

1.1 Objective of this document

This document aims to describe the tools and process to monitor operations related to the experimentations on 5GINFIRE testbed infrastructures and especially the support organization for experimenters. It also contains a report with 5GINFIRE testbed infrastructures KPIs.

1.2 Structure of this report

This report is organized as follow:

- Chapter 2 details the tools and process for experimenters support
- Chapter 3 details the tools for monitoring the testbed infrastructures in order to check the ability to host VxFs and carry experiments, including the availability of the infrastructures
- Chapter 4 reports on daily operation activities with KPIs
- Chapter 5 gives an analysis of the KPIs and provides a list of the identified next steps

2 Support & Ticketing

2.1 Introduction

Any stakeholder in 5GINFIRE should be able to request assistance in case of problem or new request. As the goal of the project is to host an experimentation environment, it is highly important to track a request/incident all along until its resolution. In order to do so, solutions like mailing lists are easy to setup but they are not sufficient in terms of tracking. In 5GINFIRE, the deployed solution has been a ticketing tool.

2.2 Requirements

Any user needs to submit requests / incidents in the ticketing tool. By "user", the following roles have been identified in accordance with the supported actors by the 5GINFIRE portal (specified in Ref[5]):

- Experimenter: can upload Experiments in terms of NSDs and request the deployment of an experiment over the 5GINFIRE infrastructure
- VxF Developer: can upload VxF archives
- Testbed provider: can register a target infrastructure
- Services administrator: responsible for the portal management
- Mentor: supervises the requirements and setup for running an experimentation

The support to the user is organized in several levels that range from general issues/request troubleshooting, to specialized troubleshooting. For 5GINFIRE project, a two level scheme has been adopted:

- Level 1: Handles the support group which is in charge of receiving the initial request and either resolve the issue or address it to appropriate group on the upper levels. The tickets are escalated by Level 1 according to their topic/nature to the appropriate assignee, with a process flow described at chapter 2.5.1.
- Level 2: Requests addressed to this layer will be managed by the Testbed Operator when it deals with an issue / request about the testbed execution environment or by the VxF Developer if it deals with the behaviour of the VxF itself. Support shall stay limited to the features provided by the VXF in the considered release.

Table 1 - 5GINFIRE ticketing process

No	Actor	Title
1	Experimenter	Request the deployment of an experiment
2	Experimenter	Raise a general issue or an issue on a testbed infrastructure or on a VxF
3	Services administrator	Analyse, filter, sort and assign the tickets raised by Experimenters
4	Testbed provider VxF Developer	Once a solution is identified by level 2 support, Testbed provider and/or VxF Developer will be in charge to apply it. The ticket will be considered as solved.
5	Experimenter	Experimenter that was the originator of a ticket shall check that the ticket is solved.
6	Services administrator	Services administrator shall close the tickets.

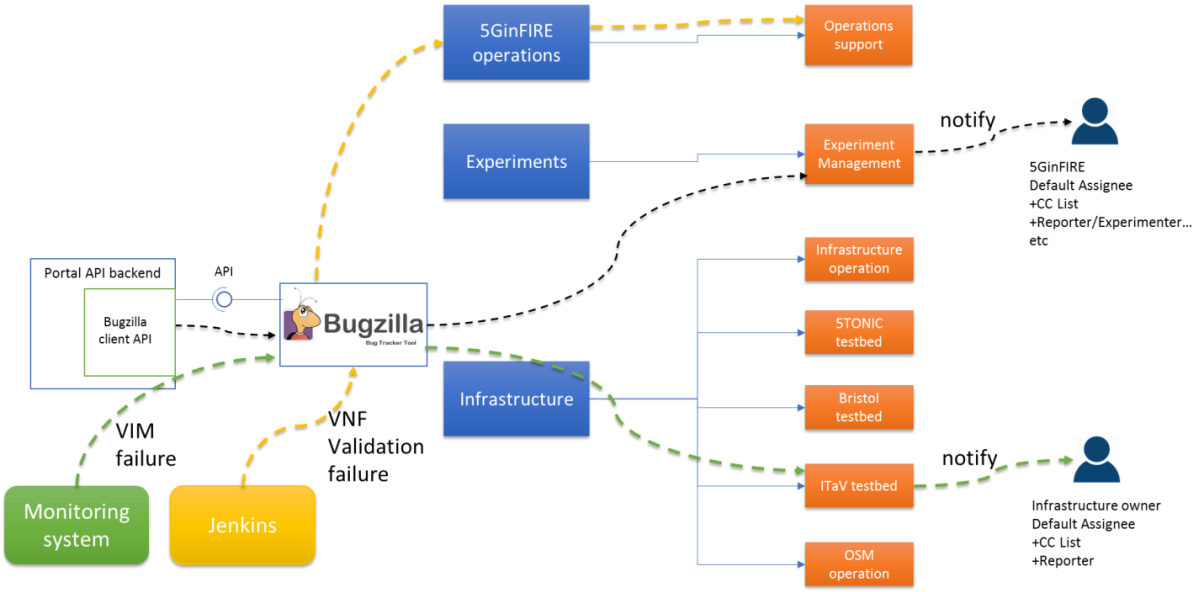
Note that support will be available on a 5 days / week basis (week-ends, bank holidays and days-off not included), at regular office open hours (9:30/12:00 - 14:00/17:30 CET/CEST Time).

For the selection of an appropriate ticketing tool, it must be taken into account that it shall provide not only a GUI for the users as mentioned above, but also an API in order to integrate the ticketing tools with the 5GINFIRE Portal and other tools for monitoring of the testbed infrastructures and the VxFs.

2.3 Architecture

The picture hereunder depicts the overall architecture and how the ticketing tool shall interact with 5GINFIRE portal and other monitoring tools. The architecture and all the tools that are interfaced with the ticketing tool are detailed in Ref[1].

Figure 1- Overall architecture



2.4 Design and implementation

The selected tool is Bugzilla (see Ref[2]). Bugzilla is mainly designed for bug-tracking inside a developers' team in order to manage issues and change requests in software products.

Bugzilla has been selected versus Jira (see Ref[3]) and GLPI (see Ref[4]) with a criteria evaluation. Jira was not selected due to the associated fee as 5GINFIRE is not an open source project.

Table 2- Ticketing tools

	Bugzilla	**Jira**	**GLPI**
Open Source	Yes	No	Yes
Licence fee	No	Free for open source projects otherwise License Fee	No
Scope	Issue management	Project and Issue management	IT assets management
Web Interface	Yes	Yes	Yes
API	Yes, RESTful API		

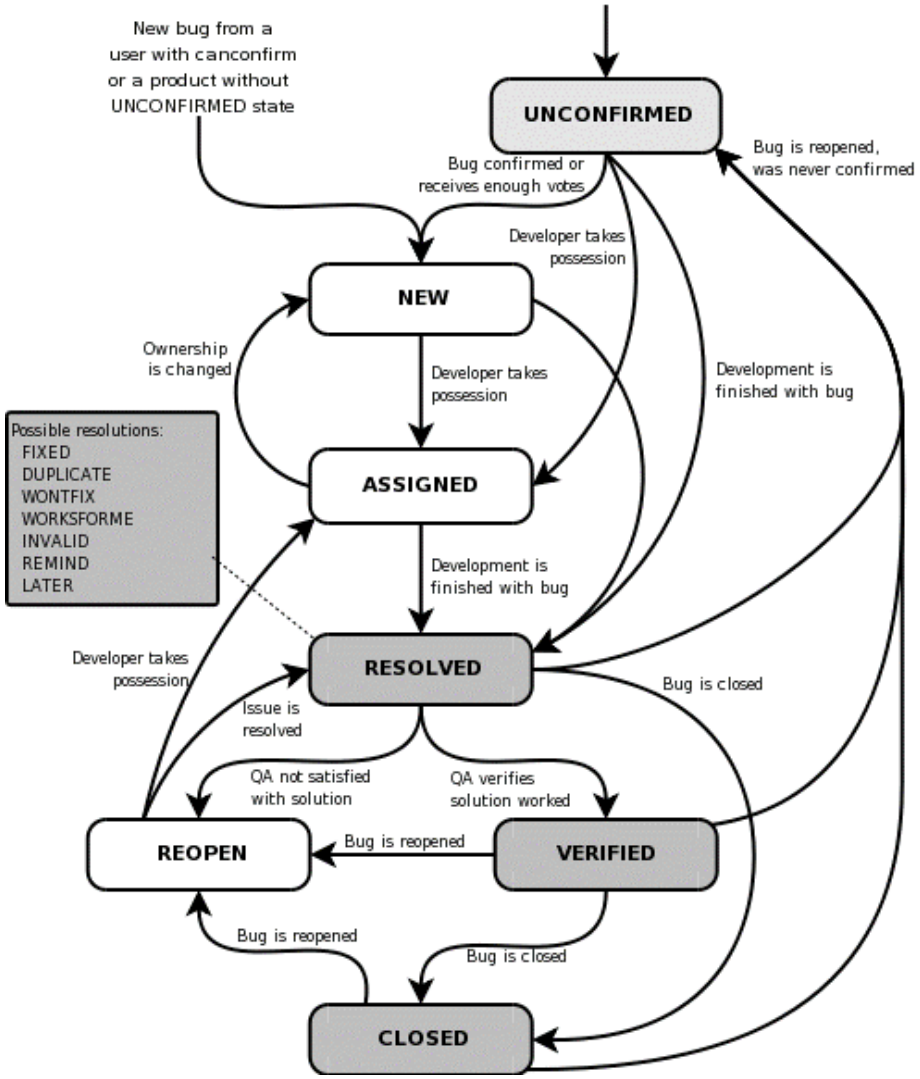
2.5 Bugzilla Deployment details

2.5.1 Ticket workflow

The workflow that has been selected by 5GINFIRE is the default one (depicted in Figure 2 below) which is used for defects in Bugzilla. In the scope of 5GINFIRE project, Bugzilla tickets are used not only for defects, but also for general requests. The workflow is equivalent and can be described as follow:

- New - A latent defect the tester enters for the first time, it is by default in the new state.
- Assigned - When the tester has logged the defect, the technical lead confirms the bug and assigns it to the corresponding developer in the development team. The defect then makes a transition to the Assigned state.
- In Progress - The developer starts addressing the bug and is currently investigating the problem. At this point, there are two possibilities of either deferring or rejecting the issue.
- Resolved - The dev team has fixed the defect, and it is ready for testing.
- Verified - The QA team has tested the error with the latest build, and the tester has confirmed the defect as fixed.
- Closed - It is the terminal state of a bug in the life cycle. The tester can close it after retesting or if he finds it as duplicate or considers as NOT a defect. In the scope of 5GINFIRE, the service administrator has also the ability to close the tickets.
- Reopened - If the bug persists even after a fix from the developer. The tester changes the status to "reopened". The bug passes through the same life cycle once again.
- Deferred - When there is no scope to address a defect in a particular bug life cycle, then it can be moved to a future release.
- Rejected - Any of the stakeholders may dismiss or discard a bug for any of the three reasons:
    - Duplicate defect,
    - Not a Defect,
    - Non-Reproducible.

Figure 2- Default workflow



2.5.2 Standard fields

In 5GINFIRE first Bugzilla deployment, the term “bug” is used for all tickets independently of whether they are actually real bugs or user requests**.**

Standard fields will be used:

- Product/Component: this two level field has been approved by project partners to describe the services provided by 5GINFIRE. “Product” field shall encompass the following list:
  - “Operations”
  - "Platform"
  - “Infrastructures”
  - “VxFs”
  - “Experiments”

  Table 3details the list for “Components” with Assignee / CC list setting (CC stands for Carbon Copy in order to notify a list of stakeholders who should be aware of a ticket even though the ticket is not assigned to them).
- Assignee/Reporter/ CC list
- Comments/Attachments

Table 3- Components table with Assignee / CC list

Product	Component	Comment	Assignee	CC list
Operations		These are reviewed during WP6 sessions to get KPIs	UoP	5GINFIRE “tickets” mailing list
Platform	Portal		UoP	Portal Owners
	Connectivity	Remote access, interconnection	UoP	5GINFIRE “testbeds” mailing list
Infrastructures	5TONIC	Including Openstack VIM, OSM MANO	UC3M	Testbed Owners
	IT-AV AUTOMOTIVE TESTBED		ITav	Testbed Owners
	Bristol Smart City Testbed		UNIVBRIS	Testbed Owners
	UFU Smart City Safety Testbed		Federal University of Uberlândia	Testbed Owners
	WINS_5G	OC-1 phase 1	Trinity College Dublin	Testbed Owners
	5G-VINO	OC-1 phase 1	University of Thessaly	Testbed Owners
	eHealth5G	OC-1 phase 1	Poznan Supercomputing and Networking Center	Testbed Owners
	PPDROne	OC-1 phase 2	INTERNET INSTITUTE	Testbed Owners
	5G Media Vertical	OC-1 phase 2	TNO	Testbed Owners
VxFs	Unifier Gateway		B-COM	VxF Owners
Experiments	RobotView5G	OC-1 phase 1	NETICTECH	Experiment Owners
	SURROGATES	OC-1 phase 1	Universidad da Murcia	Experiment Owners
	VRU-Safe	OC-1 phase 1	NKUA	Experiment Owners
	SFCLola	OC-1 phase 1	CNIT	Experiment Owners

	CAVICO	OC-1 phase 1	ITTI	Experiment Owners
	FB5G	OC-1 phase 2	Axbryd	Experiment Owners
	TelMed5G	OC-1 phase 2	medVC.eu	Experiment Owners
	5G CAGE	OC-1 phase 2	Universidad da Murcia	Experiment Owners
	DIRECTOR 5G	OC-1 phase 2	Tara Hill National Park Teo	Experiment Owners
	Sec5G	OC-1 phase 2	OneSource	Experiment Owners

2.5.3 Custom fields

In the first deployment of Bugzilla, there will be no customization and no custom fields to be added.

By default, severity field can have the following values:

- Blocker: blocks further development and/or testing work.
- Critical: Crashes, loss of data (internally, not the edit preview) in a widely used and important component.
- Major: Major loss of function in an important area.
- Normal: Default/average.
- Minor: Minor loss of function, or other problem that does not affect many people or where an easy workaround is present.
- Trivial: Cosmetic problem like misspelled words or misaligned text which does not really cause problems.
- Enhancement: Request for a new feature or change in functionality for an existing feature.

A simplified model could rely on a three severity level model to be used for notification to Testbed providers and VxF Developers in case of assistance request:

- “critical”: when it leads to the inoperability of the service and no fall-back or workaround solution is available.
- “major”: a request is “major” when it leads to a limitation of the functionalities or the performances of the service, or to the necessity to use fall-back mechanisms or workarounds.
- “minor”: a request is “minor” when it has no operational impact but leads to difficulties to operate the service.

2.6 Delivery plan

The delivery plan for the ticketing tool is depicted in Table 4. The plan for the first version was to be deployed by May 2018 in order to be ready for first interaction with experimenters selected by first open call. This target has been achieved.

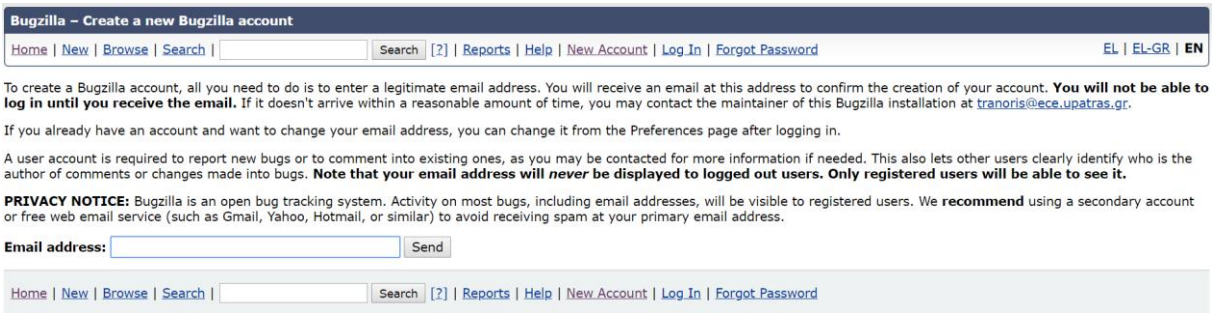
Table 4- Schedule

Version	Expected date (end of)	Description
1.0	05/2018	Ticketing tool deployed as first step with default workflow and basic use cases implemented
2.0	Q1/2019	Ticket tool based on feedback from first Open Call (if needed).

The ticketing tool is accessible from the URL specified in Ref[5].

Every new user is requested to access this URL and create an account (see Figure 3).

Figure 3- Account creation



3 Infrastructure monitoring policies

3.1 Status and KPIs to be monitored

The status monitoring and the KPIs shall reflect the status of the global 5GINFIRE infrastructure including statistics on usage, reliability and performance. So, the proposal is to organize the KPIs according to following categories:

- Infrastructure usage
- Infrastructure reliability
- Infrastructure performance

To stay in line with operation, the requirements for KPIs collection are:

- Automatic and periodic collection
- KPI per component of the infrastructure

In order to provide a health status to the users and especially experimenters, the list of monitored elements is:

- Portal admin-status {up, down, testing}, oper-status {up, down, testing},
- OSM admin-status {up, down, testing}, oper-status {up, down, testing},
- All Infrastructure admin-status {up, down, testing}, oper-status {up, down, testing},
- Interconnection State (name, type, admin-status, oper-status, last-change):
    - Between Portal and OSM
    - Between OSM towards every infrastructure (for control plane interconnection)
    - Between every infrastructure (for data plane interconnection)

The reliability is directly linked to the availability of the above mentioned elements:

- Based on ticketing tool and health-check events
- Unavailability (%)= ∑ (event opened – changed date) per element and per period, the changed date being the resolution date for a ticket which is closed
- KPI is the Availability (%) = 1 – Unavailability

The infrastructure usage KPIs are bounded to VIM resource used by the project. For each resource type, it is provided the actual quota (max) and the actual consumption (used) information. The list of available parameters is available at Ref[16].

The extraction of the aforementioned KPIs will be performed for the duration of project's 3rd period.

This list will be completed by:

- Number of instantiated VxF, deployed on testbed node
- Number of experimentation, carried on testbed node

5GINFIRE shall also cover performance indicators. From a global 5GINFIRE perspective, a first list of performance KPIs is:

- Number of Testbed nodes
- Number of VxF in catalogue
- Number of experiments per Testbed

The following performance indicators will be collected during project's 3rd period, as soon as the tools allowing the data collection are deployed:

- Infrastructure VIM performance
- Infrastructure interconnection performance

Infrastructure VIM performance is performed by collecting metrics out of the virtual machines hosting the VNFs. The metrics that can be collected for each instance are:

- CPU usage (%)
- Ram usage (%)
- Disk device byte rate (iops)
- Disk usage (bytes)
- Network interface byte rate (iops)

On the other hand, the Infrastructure interconnection performance metrics will be the following:

- Average throughput between nodes
- Average latency
- Average Jitter

3.2 Infrastructure usage

Usage information is pulled using the VIM API to request the amount of resources allocated and the ones consumed. As OSM deploys all the experiments in a single tenant on each VIM, it is quite straight forward to obtain the amount of resources booked by the project on each VIM, by obtaining the resources booked on the given tenant.

In the initial phase, a python script will be used to pull and store the metrics directly requesting the Nova API using the following OpenStack CLI command (see Ref[16] and Ref[17] for command description):

$openstack limits show –-absolute --project <Tenant>

It is being investigated whether the data can be extracted out of a new release of OSM monitoring Framework (see 3.4.1) that will be deployed end 2018 / Q1 2019, in relation with the different infrastructures.

3.3 Infrastructure reliability

5GINFIRE has developed Health Check Service (HCS), a simple service that displays the status of various services, VIMs, connectivity, components and processes of the 5GINFIRE infrastructure.

3.3.1 How it works

Components are polled (ACTIVE mode) or report their status (PASSIVE mode).

If the HCS fails to learn the status of a Component within a defined Threshold then the component is marked as FAIL/DOWN. Figure 4 shows the graphical interface of 5GINFIRE Health Status of each testbed.

Figure 4- 5GINFIRE Health Status

3.3.2 Components to be monitored and status

The following component types have been defined:

- SERVICE : a generic service, e.g. PORTAL, OSM
- PROCESS: an automated process that is successful or not. E.g. PING_PING_INSTANTIATION_TEST executed by jenkins every night
- VIM: a facility, e.g. BRISTOL
- CONNECTIVITY: a connection between components, e.g. PORTAL-OSM, OSM-ITAV, OSM-BRISTOL, etc.

A component can be in the following states:

- UP: is alive
- DOWN: is not alive
- PASS: used in PROCESS type. The process passes test within threshold
- FAIL: used in PROCESS type. The process failed test within threshold

3.3.3 Modes

There are two modes: ACTIVE and PASSIVE.

- ACTIVE mode: with active mode, the HCS polls the Component via a URL. If the URL returns 200 OK then the Component is UP. A constraint here is that the HCS must have a connectivity with the target component.
- PASSIVE mode: the PASSIVE mode can be used by components that cannot accept connections from the HCS but have internet connection (e.g. a VIM). The component reports that it is alive through a GET request to the HCS. The URL format is:

3.3.4 HCS deployment

Component providers willing to include a component in HCS shall raise a request in Bugzilla (Ref[2]), 5GINFIRE Operations/Operations Support

Admins of HCS will create the component and provides a Unique Name and *APIKEY*. The mode, the Bugzilla product, component, etc must also be agreed between component provider and HCS admins.

All the information about HCS is available in 5GINFIRE wiki (Ref[6]).

HCS is deployed at URL specified in Ref[7].

Source code is available for download or contribution at URL Ref[8].

3.3.5 Using the VNF Continuous Integration pipeline

Experimenters can submit VNFs through the portal. However, initially the submission was not controlled, that is, there was no validation of any kind on whether the submission contained a valid VNF descriptor and if it was ready to be deployed. As an upgrade, it was decided that the portal should be able to validate each VNF submission; hence, the need of having an automated testing mechanism arises.

In order to solve this problem, the inclusion of a Continuous Integration (CI) server with the Portal was evaluated as a possible solution. A CI Server is a tool that allows the automation of the development process by being able to test, build and deploy code changes submitted by the different developers of a project through a pipeline that performs those operations.

There are several CI tools, but the need was for one that was highly customizable and preferably free. The ones that stand out the most are Jenkins, Travis, Bitbucket Pipelines, CircleCI and Gitlab CI. Each one of them has the following characteristics:

- **Jenkins** is open source, free and it has very extensible features since it works based on plugins, supports multiple scripting languages and it has multiple types of pipelines. It is also platform independent and it offers an API, which enables easy integration with other applications and, in our case, with our portal. However, the configuration of the server must be done by the user and, since it is highly customizable, it may take some time to get it well configured.
- **Travis** is very easy to use, it does not need a dedicated server, supports a lot of scripting languages and offers an API. Yet, it is not free for private projects and the easy configuration has the consequence of not being really customizable.
- **Bitbucket Pipelines**, **CircleCI** and **Gitlab CI** are not so adequate for 5GINFIRE because they are only available for Bitbucket, Github and Gitlab, respectively. Since VNFs will not be stored in any of those systems, these tools are not good options.

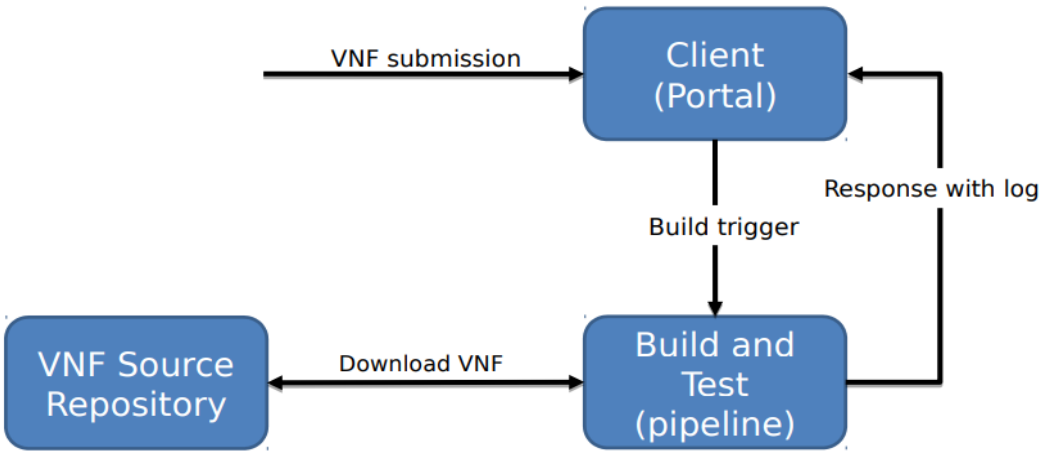
Looking at the details of each tool, 5GINFIRE project adopted Jenkins because it is open source, free, has different pipeline types (since most of the tools are restricted to source code hosting websites) and even though the server has to be configured, the specifications of 5GINFIRE project require the freedom of high customization.

A Jenkins CI server has already been installed and configured, and is available at ci.5ginfire.eu.

A pipeline was created so when a new VNF is submitted, tests are performed to draw a conclusion on whether the VNF is ready to use or not.

The pipeline workflow is illustrated in Figure 5.

Figure 5- Pipeline workflow



The pipeline has multiple stages:

1. **Download VNF -** The VNF metadata is retrieved and the package is downloaded to the Jenkins workspace. Then, the files are extracted and the descriptor is identified.
2. **Download Tests –** The tests provided by OSM are used, hence, every time the pipeline is triggered, they are fetched from their git repository.
3. **VNF Validation -** Since the OSM scripts work for their latest release, the first thing that must be done is to upgrade the descriptor to that version. Then, the descriptor is tested syntactically. Right now, this is the only test that is performed, so the validation result is sent to the portal in this stage.

5GINFIRE project is planning on improving the pipeline by adding functional tests and having a direct verification with OSM.

The target for infrastructure reliability monitoring is to integrate this pipeline with a "Test" VNF in the Health Check Service, in order to periodically test the ability of the overall 5GINFIRE infrastructure to deploy VNFs on various testbeds.

3.4 Infrastructure performance

3.4.1 OSM R4 monitoring framework

The OSM Performance Management is a new feature of OSM Release FOUR that permits monitoring and visualization of NFVI metrics, made available by the VIMs telemetry services (for example, Ceilometer - see Ref[9] - and/or Gnocchi –see Ref[10] - for OpenStack). In Release FOUR, the MON component grabs these metrics and puts them in the Kafka BUS, from where they can be read. In particular, extensions have also been added to OSM to include an optional Kafka Exporter, which:

- Reads metrics from the Kafka BUS
- Exposes them to Prometheus, so they can be stored in its database
- Presents them in Grafana (see Ref[15])

Being available from Release FOUR, it is planned to include these tools in 5GINFIRE production environment at the time of the migration from Release TWO to Release FOUR (i.e., by the end of 2018 to be ready for production for Q1 2019). For preliminary testing, a parallel environment was deployed in 5TONIC, with:

- OSM Release FOUR
- OpenStack Queens, and
- Ceilometer 10.0.2

There are numerous categories of parameters that can be monitored using this framework. The full list of available meters using Ceilometer telemetry, for example, can be found at Ref[9].

A brief summary is provided below, including those meters which are more interesting for the 5GINFIRE environment:

- OpenStack Compute
    - memory; memory.usage; memory.resident: volume of RAM allocated to the instance, used by the instance compared to the allocated, and used by the instance compared to the total available in the host machine
    - cpu; cpu_util: CPU time used and average CPU utilization
    - vcpus: number of virtual CPUs allocated to the instance
    - disk.capacity; disk.allocation; disk.usage: amount of disk that the instance can see, amount of disk occupied by the instance compared to the total in the host machine, and physical size of the image container on the host.
    - network.{incoming/outgoing}.{bytes/packets};
      network.{incoming/outgoing}.{bytes/packets}.rate;
      network.{incoming/outgoing}.packets.drop;
      network.{incoming/outgoing}.packets.error: network related counters associated with the instance
    - compute.node.cpu.{frequency; idle.time; percent; etc.}: CPU related parameters from the compute host machines.
- IPMI meters
    - hardware.ipmi.{fan; temperature; current; voltage}: notifications emitted by the bare metal services, obtained by IPMI sensors
    - hardware.ipmi.node.{power; temperature; airflow; cpu_util; mem_util; io_util}: Intel Node Manager meters besides those based on generic IPMI sensor data
- OpenStack Image Service
    - image.size: size of the uploaded image
    - image.download: image is downloaded
    - image.serve: image is served out
- OpenStack Networking
    - bandwidth: bytes through a L3 metering label

Apart from these, there is a large amount of SNMP based meters, some of them equivalent to the ones above, and meters related to SDN controllers, VPN-as-a-Service, Firewall-as-a-Service, etc. These may be investigated in the future, depending on the required capabilities.

Of course, not all the meters above have been tested already, so the final definition of what metrics will be monitored at 5GINFIRE will also have to do with the validation of the listed meters, and its ease of use. In that sense, the granularity of the measurements will be key: the granularity required to obtain meaningful bandwidth data is not the same as the one required for the number of allocated vCPUs. All this will be tuned during the first installation, and it will be possible to tweak it as needed in the future.

3.4.1.1 Approach in OSM Release FIVE

In OSM Release FIVE, apart from the NFVI metrics obtained via the VIM, it will also be possible to obtain VNF metrics, made available by OSM VCA through the Juju Metrics framework. These metrics are more related to the actual functionality of each VNF, so they can be very varied. For example, in a

video transmission VNF, an interesting metric could be the number of active users. These capabilities will be subject of further study in 5GINFIRE.

Finally, also in OSM Release FIVE, a series of metric names have been normalized. These are: *cpu_utilization; average_memory_utilization; disk_read_ops; disk_write_ops; disk_read_bytes; disk_write_bytes; packets_dropped; packets_received; packets_sent*. Hopefully, normalization will help in ensuring that VIMs beyond OpenStack are also compatible with this monitoring framework.

3.4.2 Inter-site performance measurement solution

The goal of this activity is to develop a network performance measurement solution, in order to evaluate the performance and capacity of inter-site communications that can be offered to multi-site experiments within 5GINFIRE. Due to the huge complexity of 5GINFIRE infrastructure, it is important to have a reliable tool to obtain trustworthy measurements between all sites. However, using only well-known tools such as Iperf (see Ref[11]), or any of its variants, seems to fall short to meet the necessary requirements for an infrastructure of such characteristics, as they do not provide key features like dynamical deployments or the facility to handle measurement formats.

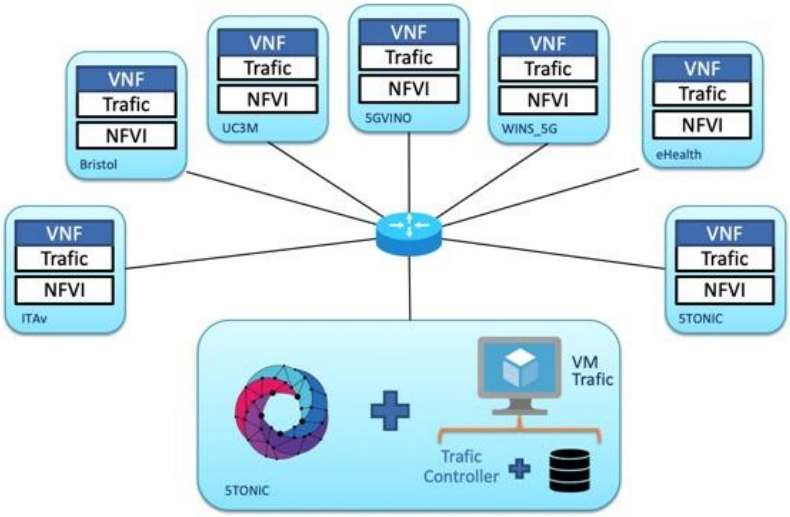
3.4.2.1 Requirements

With the aforementioned considerations, the essential requirements that this inter-site performance measurement solution should meet are the following ones:

- The solution must allow the execution of a variety of tests using realistic traffic emulation patterns.
- The solution must allow on-demand test deployments. These tests should be able to run at every moment that any measurement is needed from the network. They must also be instantiated remotely.
- The solution must allow remote data storage. All data obtained from measurements of all sites will be stored on a centralized location remotely for its extraction and further analysis when needed.
- Following the philosophy of 5GINFIRE, the solution must rely on Open-Source tools. In this respect, the development will be based on Trafic (see Ref[12]), i.e., an existing open-source traffic scheduler, developed in the MAMI project (see Ref[13]), based on Iperf3 and Influxdb (see Ref[14]). This adaptation has been done with MAMI project representatives and is a good example of H2020 cross-projects collaboration.
- Finally, the solution must allow dynamic deployments using 5GINFIRE's MANO (OSM) and be compatible with their corresponding Network Service formats.

Figure 6 shows the basic architecture designed for the solution. Each site will have one Trafic VNF. These VNFs will communicate when needed, in order to measure the performance of inter-site communications between every pair of sites. In the proposed solution, a central control unit will coordinate all the measurement process by contacting Trafic's VNFs. These ones will compose a NS, which will be deployed using the MANO platform of 5GINFIRE (based on OSM). Additionally, a remote data storage unit will collect all relevant information from the flows, resulting from the measurement process, for further analysis.

Figure 6- Architectural design of the performance measurement solution



3.4.2.2 Trafic description

Trafic's basic functioning is very simple: it relies on Iperf3 to produce a traffic flow between two different hosts (one acting as a client and another one as a server) using a set of predefined parameters like bandwidth, transport protocol, total bytes to be sent, etc. These parameters are defined inside a YAML file. These YAML files, called flows, offer a friendly readable format and very easy to configure options for every traffic flow defined.

Each file is divided in three different sections:

- A general section, where all specifications common for both client and server are defined.
- A client section, where all instantiations and configurations of a client are defined.
- A server section, where all instantiations and configurations of a server are defined.

It is important to point out that both client and server must have a copy of the same YAML file in order to match all the specifications of the flow, and allow the program to run properly.

One of the most important characteristics of Trafic is the ability to store its output measurements inside a remote *Influxdb* database. Even though Iperf3 is able to export its data to *csv* or *json* formats, they might not provide useful visualization and/or manageable options to interpret data from the emulation. However, using *Influxdb* allows retrieving data in a simpler way, and its compatibility with powerful visualization software like Grafana (see Ref[15]), makes it a far better option to use on future performance analysis for 5GINFIRE than those previously mentioned.

So far, the development has a set of predefined flow configurations that will be used to test different performance parameters over the sites:

- *Greedy*: Sends a fixed amount of bytes using the maximum bandwidth available.
- *Scavenger*: Sends a fixed amount of bytes using a predefined bandwidth.
- *Real-time audio*: A time-limited audio call emulation.
- *Real-time video*: A time-limited video call emulation.

In the future, the list of available configurations may be increased to adapt to new traffic measurement demands. Furthermore, each site could have their own set of specific flows if necessary to obtain different measurements from the ones that these pre-set configurations provide.

3.4.2.3 **Trafic functionality**

At the time of writing, the core functionality of the measurement solution has been successfully developed. The central control unit and the data storage unit have statically been deployed as virtual machines at 5TONIC. The virtual machine of the Trafic VNF has also been produced, and deployed in two datacentres available inside the 5TONIC site. With this, we have validated the functionality of the performance solution with the following test:

1. Each Trafic virtual machine initiates a *webhoo*k to listen to petitions from the controller. This *webhook* uses a previously configured URI that is always listening to a fixed URL. Once the *webhook* receives the correct URL, it triggers the action associated with it and starts the configured process.
2. Assuming that a performance measurement is needed, the controller will send an HTTP POST request to both the server and the client respectively, detailing the requested flow in its payload.
3. When both the server and the client receive the request, they will run Trafic in the corresponding mode using the YAML file described by the request.
4. The flow starts and the emulation begins, stablishing the traffic flow from the server to the client.
5. Once the process finishes (the flow is closed), the statistics of the connection are sent to the data storage unit (the *Influxdb* database).

To validate the functionality of the solution, tests have been performed to measure its reliability and accuracy. One of them was a greedy flow, sending a 100MB file through both datacentres and checking the overall throughput. This is the result, obtained using Grafana and Influxdb. Figure 7 shows the throughput using Grafana.

Figure 7- Throughput Grafana screenshot

As it can be seen on the image, the average throughput is according to the bandwidth that would be obtained through an optic fibre connected network. Therefore, the emulation works properly.

3.4.2.4 **Next steps**

Now that we have a solid base to build the complete solution, there are some steps that must be taken to further develop this idea and, in the future, incorporate it inside the project:

1. encapsulate Trafic's development as a network service;
2. deploy an instance of the Trafic VNF on each site;
3. automate the configuration of the set of flows that will initially be available at each Trafic VNF (this will be done using Juju charms and Ansible playbooks);
4. coordinate the automated test instantiation by the controller;
5. study Trafic's integration with OSM monitoring framework.

Regarding the performance KPIs to monitor, the following ones are considered:

6. average throughput among sites;

7. end to end delay of inter-site communications;
8. delay jitter for network communications established among sites.

The average figures of these parameters will be estimated in different times of the day (e.g., every two hours), to evaluate the variations of the performance over time, and expose this information to experimenters.

4 H2 2018 statistics

4.1 Q3 2018 KPIs

Table 5– Q3 2018 KPIs

KPI	Values	Comments
Performance KPIs		
Testbeds up and running	7	4 core partners (5TONIC, IT-AV AUTOMOTIVE TESTBED, Bristol Smart City Testbed, UFU Testbed) + eHealth Testbed + NITOS testbed + Iris Testbed
Running experimentations	5	SURROGATES, CAVICO, SFCLola, RobotView5G, VRU-Safe
Registered Users	61	Extracted out of ticketing tool
Reliability KPIs		
Unique tickets Submitters	20	Extracted out of ticketing tool
New requests	232	Extracted out of ticketing tool
Remaining Open requests	20	Extracted out of ticketing tool
Remaining Open blocker/critical requests	3	Extracted out of ticketing tool
Resolved / Verified requests	216	Extracted out of ticketing tool
Average time to solve a ticket (days)	11,65	Extracted out of ticketing tool
Platform	**Availability**	
Portal	100,00%	Calculated after extraction of ticketing tool
Connectivity with		
OSM	100,00%	Calculated
5G-VINO	93,47%	Calculated
5TONIC testbed	100,00%	Calculated
Bristol testbed	99,63%	Calculated
eHealth5G	52,71%	Calculated
ITaV testbed	53,71%	Calculated
UFU Testbed	86,88%	Calculated
WINS_5G	28,88%	Calculated
Infrastructures		
5G-VINO	96,04%	Calculated
5TONIC testbed	100,00%	Calculated

Bristol testbed	100,00%	Calculated
eHealth5G	84,56%	Calculated
ITaV testbed	68,70%	Calculated
UFU Testbed	Not available(*)	Calculated
WINS_5G	23,20%	Calculated

Q3 period was still a transition period and to apply the process for acknowledging and fixing the DOWN status events. More specifically, 5G-VINO, eHealth5G and WINS_5G joined the 5GINFIRE infrastructures in Q2 2018 and Q3 2018 was dedicated to setup and interconnect those testbeds. For UFU Testbed, the sum of the unavailability periods was greater than the observation period, due to the fact that the acknowledgement of the DOWN status events has not been applied during this period.

4.2 Q4 2018 KPIs

Table 6– Q4 2018 KPIs

KPIs	Values	Comments
Performance KPIs		
Testbeds up and running	7	4 core partners (5TONIC, IT-AV AUTOMOTIVE TESTBED, Bristol Smart City Testbed, UFU Testbed) + eHealth Testbed + NITOS testbed + Iris Testbed
Running experimentations	5	SURROGATES, CAVICO, SFCLola, RobotView5G, VRU-Safe
Registered Users	61	Extracted out of ticketing tool
Reliability KPIs		
Unique tickets Submitters	21	Extracted out of ticketing tool
New requests	180	Extracted out of ticketing too
Remaining Open requests	10	Extracted out of ticketing tool
Remaining Open blocker/critical requests	1	Extracted out of ticketing tool
Resolved / Verified requests	203	Extracted out of ticketing tool
Average time to solve a ticket	3,2	Extracted out of ticketing tool
Platform	**Availability**	
Portal	84,5%	Calculated after extraction of ticketing tool
Connectivity with		
OSM	94,90%	Calculated
5G-VINO	28,60%	Calculated
5TONIC testbed	94,45%	Calculated

Bristol testbed	72,01%	Calculated
eHealth5G	94,95%	Calculated
ITaV testbed	63,37%	Calculated
UFU Testbed	36,40%	Calculated
WINS_5G	71,84%	Calculated
Infrastructures		
5G-VINO	99,55%	Calculated
5TONIC testbed	96,10%	Calculated
Bristol testbed	100,00%	Calculated
eHealth5G	100,00%	Calculated
ITaV testbed	100,00%	Calculated
UFU Testbed	44,71%	Calculated
WINS_5G	24,34%	Calculated

The Q4 period was stopped the 10th of December 2018 in order to process the data for this deliverable due for the end of December.

5 Conclusion and next steps

5.1 Period 2 KPIs analysis

Thanks to the tooling which has been put in place before the first open call, it has been possible to connect new infrastructures and host experiments with a good level of support from the 5GINFIRE operation team.

The KPIs shows:

- The ticketing tool is used by the 5GINFIRE community including the users and the operation team
- The process to manage the tickets is applied and the average time to solve a ticket has decreased between Q3 and Q4 2018, down to 3,2 days
- The number of open tickets is kept under control and is followed during regular WP6 meetings
- With Portal availability close to 85% and OSM connected with many infrastructures like 5TONIC, Bristol, IT-AV and eHealth5G near or above 95%, 5GINFIRE succeeded to build a solid framework to host experimentations.
- Nevertheless, there are some opportunities for various testbeds to improve their availability associated with their connectivity with OSM. This will be a target for next period.

5.2 Next steps

5.2.1 Ticketing

Currently, the data are exported at CSV format and then post-processed with Microsoft excel in order to get the KPIs. 5GINFIRE project will investigate if the selected tool for ticketing (i.e. bugzilla) can be completed with plugins in order to enable automatic reports.

The 3 levels severity model will not be natively implemented in Bugzilla , but rather the following mapping will be considered at the post-processing stage:

- “critical” = Blocker + Critical
- “major” = Major + Normal
- “minor” = Minor + Trivial

5.2.2 Event & Log aggregation

HealthCheck Service provides an instantaneous status of the infrastructure. To demonstrate the reliability of 5GINFIRE infrastructure, an historical view of the events and logs including the availability/unavailability data is foreseen for the last period of the project.

To achieve this, 5GINFIRE has developed the 5GINFIRE Central Logging Service, which is a simple service that collects various logging information from components of the 5GINFIRE infrastructure. It enables to retrieve historical data, usage and actions of various components.

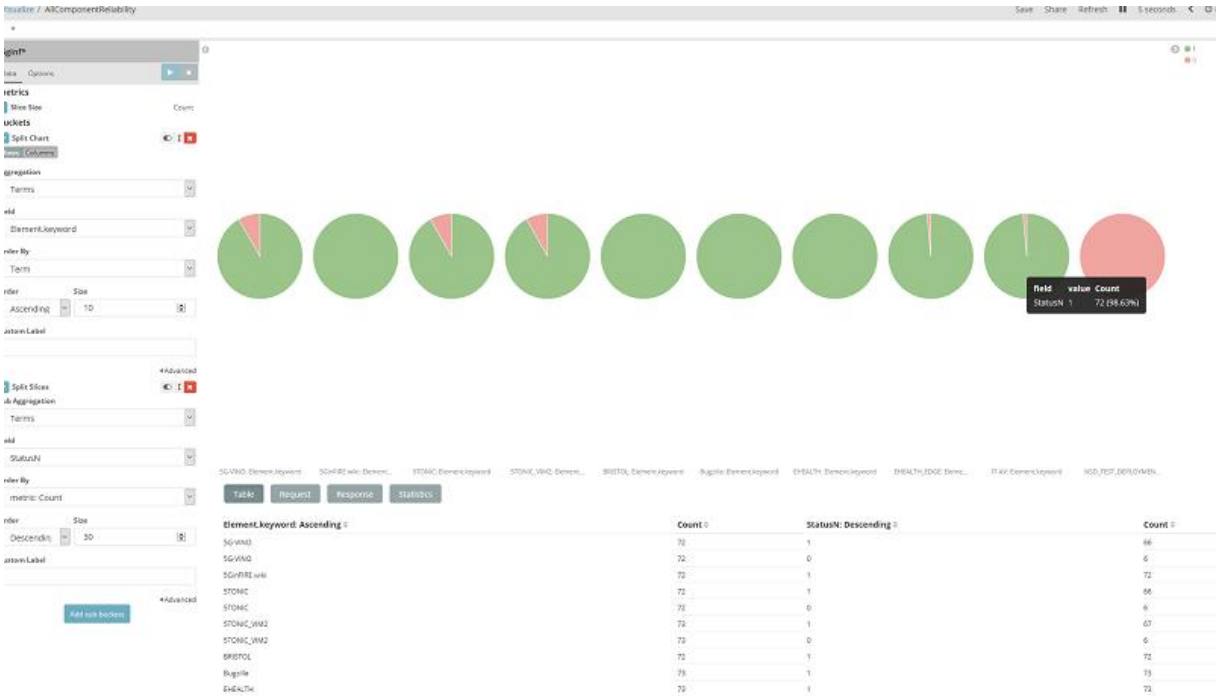
The service allows all registered Components to POST a logging message through the 5GINFIRE Healthcheck service (HCS) and this message is stored in an Elasticsearch cluster and later on examined via a Kibana service as the next figure shows.

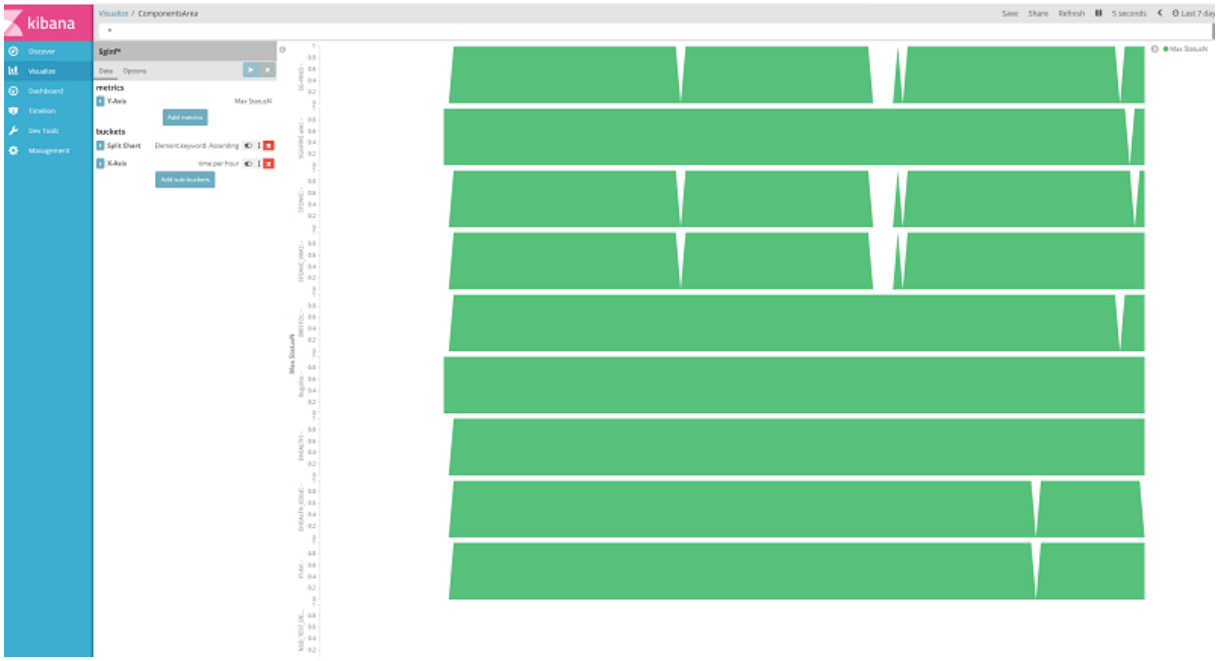
Figure 8- 5GINFIRE Health Status historical data collection

For example, all the components' activity will be stored in an elasticsearch cluster for up/down time analysis according to the architecture depicted in the figure below.

This event & log aggregation has already been prototyped and some screenshots of the service are provided hereunder.

Figure 9- 5GINFIRE Health Status historical data prototype screenshots





The target is to deploy this data collection tool by beginning 2019, for the beginning of period 3.

5.2.3 KPIs

In order to improve the monitoring of the 5GINFIRE infrastructure, the project has identified improvement steps for every category of KPIs.

- Reliability KPIs: deployment of the CI pipeline for monitoring the ability to deploy VNFs from central OSM orchestrator towards testbeds
- Usage KPIs collection from remote VIMs with elasticsearch and Kibana. It should be stated if such statistics could be provided by OpenVIM, which is the current VIM for 5G-VINO. Otherwise, a migration of 5G-VINO to OpenStack VIM could be planned beginning H2 2019.
- Performance KPIs, with the deployment of performance monitoring tools specified by WP4. The target is to have performance KPIs at least for H2 2019.

6 References

[1] Christos Tranoris, 5GINFIRE Project, D3.1 - 5G Experimentation portal, tools and middleware

[2] Bugzilla, a defect-tracking system https://www.bugzilla.org/

[3] Jira, a software development tool used by agile teams https://www.atlassian.com/software/jira

[4] GLPI, an ITSM software tool http://glpi-project.org/

[5] https://portal.5ginfire.eu/bugzilla/

[6] http://wiki.5ginfire.eu/hcservice/usage

[7] http://status.5ginfire.eu/

[8] https://github.com/5GinFIRE/eu.5ginfire.healthcheck

[9] The OpenStack Telemetry service: https://docs.openstack.org/ceilometer/queens/admin/telemetry-measurements.html

[10] Gnocchi, metric as a service: https://gnocchi.xyz/

[11] iPerf - The TCP, UDP and SCTP network bandwidth measurement tool (last access on Dec. 2018) https://iperf.fr/

[12] Trafic: A traffic mix generator based on iperf3 (last access on Dec. 2018): https://github.com/mami-project/trafic

[13] MAMI H2020 project: https://mami-project.eu/

[14] Influxdb: A database based on timeseries to store all of its contents (last access on Dec.2018): https://www.influxdata.com/time-series-platform/influxdb/

[15] Grafana: An open-source software for time-series analytics (last access on Dec. 2018): https://grafana.com

[16] Nova: Quotas https://docs.openstack.org/nova/queens/admin/quotas.html

[17] Nova: Limits https://docs.openstack.org/python-openstackclient/queens/cli/command-objects/limits.html