Programmable edge-to-cloud virtualization fabric for
the 5G Media industry

APIs and Tools for Operation Support

Work Package: WP2 – Architecture, Analysis and Tools
Lead partner: ENG
Author(s): Francesco Iadanza[ENG], David Breitgand [IBM], Avi Weit
[IBM], Takis Athanasoulis [SILO], Stamatia Rizou [SILO], Ugur
Acar [NET]
Delivery date (DoA): November 30th, 2017
Actual delivery date: December 1st, 2017
Dissemination level: Public
Version number: 1.0 Status: Final
Grant Agreement N°: 761699
Project Acronym: 5G-MEDIA
Project Title: Programmable edge-to-cloud virtualization fabric for the 5G
Media industry
Instrument: IA
Call identifier: H2020-ICT-2016-2
Topic: ICT-08-2017, 5G PPP Convergent Technologies, Strand 2:
Flexible network applications
Start date of the project: June 1st, 2017
Duration: 30 months

Project co-funded by the European Commission under the
Horizon 2020 Programme.
## Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Who</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Sept. 1&lt;sup&gt;st&lt;/sup&gt; 2017</td>
<td>ENG</td>
<td>TOC and links to the tools available</td>
</tr>
<tr>
<td>0.2</td>
<td>Sept. 15&lt;sup&gt;th&lt;/sup&gt; 2017</td>
<td>ENG</td>
<td>Review; added selection criteria on tools</td>
</tr>
<tr>
<td>0.3</td>
<td>Sept. 29&lt;sup&gt;th&lt;/sup&gt; 2017</td>
<td>ENG</td>
<td>Review; added DevOps and configuration management tools</td>
</tr>
<tr>
<td>0.4</td>
<td>Oct. 13&lt;sup&gt;th&lt;/sup&gt; 2017</td>
<td>ENG</td>
<td>Review</td>
</tr>
<tr>
<td>0.5</td>
<td>Oct. 28&lt;sup&gt;th&lt;/sup&gt; 2017</td>
<td>ENG</td>
<td>Review and integration of contributions</td>
</tr>
<tr>
<td>0.6</td>
<td>Nov. 16&lt;sup&gt;th&lt;/sup&gt; 2017</td>
<td>ENG</td>
<td>Final review and integration</td>
</tr>
<tr>
<td>1.0</td>
<td>Dec. 1&lt;sup&gt;st&lt;/sup&gt; 2017</td>
<td>ENG</td>
<td>Final version for submission</td>
</tr>
</tbody>
</table>

## Quality Control

<table>
<thead>
<tr>
<th>Role</th>
<th>Date</th>
<th>Who</th>
<th>Approved/Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal reviewer</td>
<td>01/11/2017</td>
<td>SILO</td>
<td></td>
</tr>
<tr>
<td>Internal reviewer</td>
<td>16/11/2017</td>
<td>NET</td>
<td></td>
</tr>
</tbody>
</table>
Disclaimer

This document may contain material that is copyright of certain 5G-MEDIA project beneficiaries, and may not be reproduced or copied without permission. The commercial use of any information contained in this document may require a license from the proprietor of that information. The 5G-MEDIA project is part of the European Community's Horizon 2020 Program for research and development and is as such funded by the European Commission. All information in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability. For the avoidance of all doubts, the European Commission has no liability with respect to this document, which is merely representing the authors’ view.

The 5G-MEDIA Consortium is the following:

<table>
<thead>
<tr>
<th>Participant number</th>
<th>Participant organisation name</th>
<th>Short name</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>ENGINEERING – INGEGNERIA INFORMATICA SPA</td>
<td>ENG</td>
<td>Italy</td>
</tr>
<tr>
<td>02</td>
<td>IBM ISRAEL - SCIENCE AND TECHNOLOGY LTD</td>
<td>IBM</td>
<td>Israel</td>
</tr>
<tr>
<td>03</td>
<td>SINGULARLOGIC ANONYMI ETAIREIA PLIROFORIAKON SYSTIMATON KAI EFARMOGON PLIROFORIKIS</td>
<td>SILO</td>
<td>Greece</td>
</tr>
<tr>
<td>04</td>
<td>HELLENIC TELECOMMUNICATIONS ORGANIZATION S.A. - OTE AE (ORGANISMOS TILEPIKIONION TIS ELLADOS OTE AE)</td>
<td>OTE</td>
<td>Greece</td>
</tr>
<tr>
<td>05</td>
<td>CORPORACION DE RADIO Y TELEVISION ESPANOLA SA</td>
<td>RTVE</td>
<td>Spain</td>
</tr>
<tr>
<td>06</td>
<td>UNIVERSITY COLLEGE LONDON</td>
<td>UCL</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>07</td>
<td>TELEFONICA INVESTIGACION Y DESARROLLO SA</td>
<td>TID</td>
<td>Spain</td>
</tr>
<tr>
<td>08</td>
<td>UNIVERSIDAD POLITECNICA DE MADRID</td>
<td>UPM</td>
<td>Spain</td>
</tr>
<tr>
<td>09</td>
<td>INSTITUT FUER RUNDFUNKTECHNIK GMB</td>
<td>IRT</td>
<td>Germany</td>
</tr>
<tr>
<td>10</td>
<td>NEXTWORKS</td>
<td>NXW</td>
<td>Italy</td>
</tr>
<tr>
<td>11</td>
<td>ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS</td>
<td>CERTH</td>
<td>Greece</td>
</tr>
<tr>
<td>12</td>
<td>NETAS TELEKOMUNIKASYON ANONIM SIRKETI</td>
<td>NET</td>
<td>Turkey</td>
</tr>
<tr>
<td>13</td>
<td>INTERINNOV SAS</td>
<td>IINV</td>
<td>France</td>
</tr>
<tr>
<td>14</td>
<td>BITTUBES GMB</td>
<td>BIT</td>
<td>Germany</td>
</tr>
</tbody>
</table>
# Table of Contents

**EXECUTIVE SUMMARY** ................................................................................................................... 9

1. **INTRODUCTION TO CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY** .......... 10
   1.1.  *The DevOps Approach* ............................................................................................................. 10
   1.2.  *Continuous Integration and Continuous Delivery Tools Categories* ................................... 11

2. **TOOLS PROVIDED TO THE PROJECT** ......................................................................................... 14
   2.1.  *Project Portal* .......................................................................................................................... 16
       2.1.1.  *Description of the tool* ....................................................................................................... 16
       2.1.2.  *Tool available for the project, configuration and guidelines* ........................................... 17
   2.2.  *Version Control System* .......................................................................................................... 18
       2.2.1.  *Description of the tools, comparison and selection among alternatives* ...................... 18
       2.2.2.  *Tool available for the project, configuration and guidelines* ........................................... 22
   2.3.  *Issue Tracking System* ............................................................................................................ 23
       2.3.1.  *Description of the tools, comparison and selection among alternatives* ...................... 23
       2.3.2.  *Tool available for the project, configuration and guidelines* ........................................... 23
   2.4.  *Automation Server* ................................................................................................................... 25
       2.4.1.  *Description of the tools, comparison and selection among alternatives* ...................... 25
       2.4.2.  *Tool available for the project, configuration and guidelines* ........................................... 27
   2.5.  *Binary Repository Manager* .................................................................................................... 28
       2.5.1.  *Description of the tools, comparison and selection among alternatives* ...................... 28
       2.5.2.  *Configuration and guidelines for the project* .................................................................... 29
   2.6.  *Code Quality Service* .............................................................................................................. 30
       2.6.1.  *Description of the tools, comparison and selection among alternatives* ...................... 30
       2.6.2.  *Configuration and guidelines for the project* .................................................................... 30
   2.7.  *Containerization Tools* .......................................................................................................... 31
       2.7.1.  *Description of the tools, comparison and selection among alternatives* ...................... 31
       2.7.2.  *Configuration and guidelines for the project* .................................................................... 33
   2.8.  *Configuration Management Tools* ........................................................................................... 34
       2.8.1.  *Description of the tools, comparison and selection among alternatives* ...................... 34
       2.8.2.  *Configuration and guidelines for the project* .................................................................... 35
   2.9.  *Serverless Computing: OpenWhisk* ......................................................................................... 35
       2.9.1.  *Description of the tool* ....................................................................................................... 35
       2.9.2.  *Configuration and guidelines for the project* .................................................................... 38

3. **HIGH-LEVEL CI/CD PIPELINE FOR THE PROJECT** ................................................................. 39
List of Figures

Figure 1 - Dev & Ops collaboration (source: MartinFowler.com ) ............................................. 11
Figure 2 - ENG Identity Provider ................................................................................................. 14
Figure 3 - ENG services login ...................................................................................................... 15
Figure 4 - change password for external users ........................................................................... 15
Figure 5 - change password for internal (ENG) users ................................................................. 15
Figure 6 - tools technical guide referenced on Wiki ................................................................. 16
Figure 7 - Alfresco document repository for 5G-MEDIA ......................................................... 17
Figure 8 - Wiki service customized for 5G-MEDIA ................................................................. 18
Figure 9 - VCS usage trends (Source: Google Trends [Google6]) ........................................... 19
Figure 10 - Git flow example (source: Gitlab.com ) ................................................................. 20
Figure 11 - simple GitHub flow (source: Gitlab ) ...................................................................... 21
Figure 12 - "rebase" operation in Git (source: Atlassian ) ......................................................... 22
Figure 13 - Gitlab service for 5G-MEDIA .................................................................................. 22
Figure 14 - JIRA activity/bug workflow for the project ............................................................. 24
Figure 15 - JIRA service for 5G-MEDIA ..................................................................................... 24
Figure 16 - Jira component management to classify issues ........................................................ 25
Figure 17 - best automation tools for enterprise compared (Source: G2 Crowd Grid [4]) ... 26
Figure 18 - Jenkins features compared with other open source tools (Source: Wikipedia [Wikipedia7]) .................................................................................................................. 27
Figure 19 - Jenkins service for 5G-MEDIA ................................................................................ 27
Figure 20 - Repository Manager ranking (Source: Github [Repository8]) .............................. 29
Figure 21 - Nexus service for 5G-MEDIA ................................................................................... 30
Figure 22 - SonarQube service for 5G-MEDIA ......................................................................... 31
Figure 23 - Docker growth (Source: coscale.com, from DockerCon 2016 [Docker9]) ......... 32
Figure 24 - Docker adoption in production environments (Source: Datadog [Datatog10]) ................................................................. 32
Figure 25 - OpenWhisk Programming Mode ............................................................................. 37
Figure 26 - OpenWhisk Architecture ......................................................................................... 38
Figure 27 - high-level CI/CD pipeline based on Docker for 5G-MEDIA project .................. 40
List of Tables

Table 1 - list of the selected tools for the 5G-MEDIA project .......................................................... 40
## Definitions and acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>5G-PPP</td>
<td>5G Infrastructure Public Private Partnership</td>
</tr>
<tr>
<td>CD</td>
<td>Continuous Delivery</td>
</tr>
<tr>
<td>CI</td>
<td>Continuous Integration</td>
</tr>
<tr>
<td>CLI</td>
<td>Command Line Interface</td>
</tr>
<tr>
<td>FaaS</td>
<td>Function as a Service</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>IaC</td>
<td>Infrastructure as Code</td>
</tr>
<tr>
<td>IaaS</td>
<td>Infrastructure as a Service</td>
</tr>
<tr>
<td>OW</td>
<td>OpenWhisk</td>
</tr>
<tr>
<td>SSO</td>
<td>Single Sign On</td>
</tr>
<tr>
<td>VCS</td>
<td>Version Control System</td>
</tr>
<tr>
<td>VIM</td>
<td>Virtual Infrastructure Manager</td>
</tr>
<tr>
<td>WP</td>
<td>Work package</td>
</tr>
</tbody>
</table>
Executive summary

This document describes the tools provided to the 5G-MEDIA project to support Continuous Integration (CI) and Continuous Delivery (CD).

A brief description is provided for each tool, together with the rationale adopted for the selection among the existing alternatives. Together with the reference to the official documentation for each tool, this document also contains the guidelines adopted in the 5G-MEDIA project and the references to the project Wiki for the technical details that may change frequently.

The final section contains the description of a high-level CI/CD pipeline for the project relying on such tools.
1. Introduction to Continuous Integration and Continuous Delivery

1.1. The DevOps approach

The term “DevOps” represents a software engineering practice born, like many others, from the common experiences of communities of software developers and system administrators for operation.

Such communities have been modifying the main essence of “DevOps” over time and so the Wikipedia definition [Wikipedia1] that has come along, that in September 2017 changed to “a practice that advocates automation and monitoring at all steps of software construction, from integration, testing, releasing to deployment and infrastructure management ... in alignment with business objectives”, thus focusing on automation and monitoring, yet recognising a key role to effective communication among teams.

Since its first definition during the 2008 Agile Toronto conference \(^1\), DevOps has always been considered as strictly related to the Agile movement especially for the communication principle of the Agile manifesto\(^2\) “individuals and interactions over processes and tools”.

So, one side DevOps advocates the continuous monitoring of the communication process and coordination and focuses on soft skills, on the other is about the tools to speed up software development, deploying, testing.

A measurable advantage of DevOps is an efficient communication among different teams, that in software development and operation can be declined into sharing an overall view of the system and the related responsibilities for each team, the production of feedbacks to capture issues and potential improvements to application development workflow.

An example of such optimization using the DevOps methodology applied to 5G-MEDIA area of interest is the one proposed by Nokia for the deployment of Virtual Network Functions [VNF19], focused on the validation of the resources to be allocated by the virtual infrastructure before the deployment and the subsequent verification of possible issues.

The definition of such methodology is domain specific and beyond the scope of this document; nevertheless, some of the tools described in the next section can be used to enhance communication at different levels, from document sharing and collaborative editing to activity and issue tracking, as described later, to encourage the collaboration among the project teams, as shown in the Figure 1.

\(^1\) DevOps in 2008 Agile Toronto conference, \(\text{http://itrevolution.com/the-history-of-devops/}\)

\(^2\) The Agile Manifesto, \(\text{http://agilemanifesto.org/}\)
On the other hand, the main advantage of DevOps is about the reduction of software development cycle time, through the automation and monitoring of repetitive tasks. Again, this should be considered as the result of the collaboration among teams, that shares the responsibility to look after the system over its lifetime and highlight new requirements to the DevOps.

This document just focuses on the selection of the most appropriate set of tools able to support software development and provide those automation features that DevOps advocates, based on Continuous Integration (CI), Continuous Delivery (CD) and comprehensive of continuous monitoring of the expected results.

1.2. Continuous Integration and Continuous Delivery tools categories

The two key components of the DevOps approach applied to software development are CI and CD that focus on minimizing the elapsed time from the integration of software requirements to the release, the so called “cycle time”.

More specifically, CI enables developers to regularly merge their code changes into a central repository and have automated builds and unit tests, to address bugs quickly and improve software quality.

CD is the following step of CI, that enables the delivery of an application to the Quality Assurance (QA) team for system and integration testing, then to the operations team for the release in production. This does not mean the automatic delivery in production, but more the

capability to deploy an entire application or a component with a minimal amount of human intervention in order to execute tests, as from the famous statement from Carl Caum “continuous delivery means that every change is proven to be deployable at any time” 5.

Among the most recognized leaders in the software development methodologies and one of the authors of the Agile Manifesto, Martin Fowler’s best practices for CI [Fowler13] help to define the most appropriated tools categories that should be used for such methodology, that he advises to be:

- a tool for source version control;
- a tool for automatic building, dependency checking and automatic testing;
- a tool to keep tracking of the issues in order to fix them immediately and, more in general, one for software quality checking to allow the continuous improvement promoted by the Agile methodology (e.g. SCRUM with sprint retrospective);
- a tool for automatic deployment.

The same author defines the CD as a process based on a deployment pipeline [Fowler14] having, above all:

- the capability to deploy any version of software to any environment, on demand;
- the capability to report continuously feedbacks on the readiness of the system.

Such deployment pipeline describes a typical workflow with the steps that source code goes through to make its way to production, and involves build, test, staging, acceptance tests, deploy to production, with an overall continuous monitoring and reporting for bug issues.

It is based on a binary repository to store application components to be deployed on staging and allow integration and quality acceptance tests before being deployed on production environment.

This phase is part of the CD and can be done by specific scripts for packaging, deploying and changing configuration often called as configuration management tools.

The capability to deploy on “any environment” requires platform provisioning and configuration management in order to create, maintain and tear down a complete test environment automatically, while “on demand” is usually meant as a “pull system” to enable the deployment with few interactions from the developers, for example through as a Graphical User Interface (GUI) that allows to configure the preferred test environment.

Most important, supporting “any environment” is enabled by virtual infrastructures, that provides an abstraction over hardware details and allows to simulate different environments for testing and production. Such infrastructures may rely on containerization as a more efficient way to pack more applications into a single physical server than a virtual machine can do and follows the principle of optimization that DevOps advocates.

5 Carl Caum (CEO of Puppet) about Continuous Delivery, https://tinyurl.com/zsoenks
From the definition of deployment pipeline by Martin Fowler [Fowler15], the most appropriate tools categories for CD are:

- a repository for the binary distribution,
- tools for configuration management for deploying and testing applications on the environments provided by some platform provisioning services,
- reports to provide feedbacks for retrospective analysis.

These tools, together with those more general for effective communication among teams, will be described in the next section.

While virtualization and containerization techniques provide a higher level of abstraction over the existing architecture and ease the process of provisioning and creating different test environments, the developer has still the burden of managing such resources even though Infrastructure as Code (IaC)\(^6\) simplifies this process reducing the human intervention with a bunch of application specific configuration management scripts.

This is particularly evident when doing performance testing or even in production, when the need for orchestration of resources to simulate or satisfy different computation loads requires a continuous monitoring from the system administrators and their intervention to change the system configuration.

The serverless computing approach\(^7\) simplifies such management, as it allows to build and run applications without thinking about the servers at all, as the main principle is that provisioning, scaling and management is taken care automatically by a set of preconfigured servers in the cloud in a transparent way for the developers.

This mean a reduced effort in the CI and CD pipeline, as there is still the need for a central source code repository, building and testing, but they often rely on tools that are already provided by the serverless architecture vendor through a SDK.

The serverless computing approach is rather recent, yet some tools exist to abstract from the specific vendors and should allow to provide a common base to build CI/CD integration regardless of vendor specific SDKs.

OpenWhisk\(^8\) is an open source Function as a Service (FaaS) technology, in which the main primitive is an action. An action is basically the code that is executed, it does some typically short-lived task, then it terminates and becomes dormant. The actions repository is managed in centralized way in a cloud. The repository contains actions metadata and code. In addition, actions, as well as other assets, such as triggers, rules, packages and combiners, can be annotated by an arbitrary number of user defined annotations. Presently, explicit version control is not supported out of the box by OpenWhisk for the user developed assets. However, annotations can be used as hooks for the version control.


\(^8\) OpenWhisk official website, [https://openwhisk.apache.org/](https://openwhisk.apache.org/)
One potential gap that exists in OpenWhisk (and other FaaS technologies) is absence of a gradual rollout and quick rollback of features at the function level. These capabilities are expected in the future and some of these shortcomings can be addressed in the 5G-MEDIA project.

2. Tools provided to the project

This section describes the tools selected so far for the 5G-MEDIA project, although many changes along the project time life are still possible and will be reported in the Wiki of the project as a result of the task “T2.3 5G-MEDIA Operation Support Tools”.

As the 5G-MEDIA project spans over a period of more than two years, the main rationale for the selection has been the sustainability, with the selection (in most of the cases) of open source tools with a strong community, capable to offer a valuable support.

Most of the tools listed below are hosted on ENG cloud infrastructure, with access granted according to a list of authorized users shared with ENG and updated over time. Any request regarding new users as well as configuration changes should be managed opening an issue to ENG using JIRA in order to keep track of the operations.

Access to ENG services is allowed after successful login using the same credentials but as no Single Sign On (SSO) is available, a logon is necessary every time a new service is required. It is necessary to select the correct Identity Provider between internal users (ENG) and external ones. As the login panel is not completely localized in English, the former will be “Gruppo Engineering”, the latter “Utenti esterni”, as shown in the Figure 2 and then enter the credentials as shown in the Figure 3.

![ENG Identity Provider](image)

*Figure 2 - ENG Identity Provider*
Temporary passwords have been provided by email; to change it external user must click the “Change Password” tab on the top left as shown in the Figure 4, again with some not English localized content, with textboxes for the old password (“vecchia”) and the new one (“nuova”). On the other hand, the change password feature shown in the Figure 5 is working only for internal (ENG) users.

Finally, a technical guideline to configure the tools is available on the Wiki (“API and Tools for Operation - technical guide”, see Figure 6) and has been separated by this document and contains all the examples and configurations that are going to change over the project time life.
API and Tools for Operation - technical guide

Technical guidelines for the 5G-MEDIA tools described in the deliverable D2.1

1. General guidelines

All the tools described below are available with the same credentials provided by Engineering to access the project portal.

Any change to the tools configuration should be requested to Engineering using the JIRA tool (after the official release of the D2.1).

2. Tools guidelines

Jira: configured with the default "Jira Agile SCRUM" workflow.

Jenkins: configured by default to support Java and Maven projects.

The following snippet must be added to the POM file for Maven projects in order to use the Nexus repository:

```xml
    <project>
        <distributionManagement>
            <repository>
                <id>5G-releases</id>
                <url>https://production.org/alfresco/content/repos/5G-MEDIA/</url>
            </repository>
        </distributionManagement>
        <!-- use the following if you're not using a snapshot version. -->
        <repository>
            <id>5G-releases</id>
            <url>https://production.org/alfresco/content/repos/5G-MEDIA/</url>
        </repository>
    </project>
```

Figure 6 - tools technical guide referenced on Wiki

2.1. Project portal

2.1.1. Description of the tool

The project portal contains the internal documentation, integrates an ad-hoc instance of Alfresco for document management and a Wiki for collaborative content management.

Alfresco is an Enterprise Content Management system and the official documentation is available at the URL http://tinyurl.com/5gmedia-alfresco.

The main features are document management and versioning, tools for the review process, fine grained access control, as described in the official documentation available at the URL https://www.alfresco.com/capabilities/document-management.
Wiki is a collaborative content website included in the project portal provided by ENG basic tools and it is based on Liferay.

The main features are add/modify/remove content, online editor, preview, access management, feeds and content versioning.

2.1.2. Tool available for the project, configuration and guidelines

Alfresco is used in 5G-MEDIA to store the project consolidated documents. The version installed is the Community Edition 4.2.0.

To have Alfresco documents automatically synchronized on a local PC, CmisSync Free⁹ can be used, configured with the URL https://production.eng.it and the username and password provided by ENG.

Wiki is used for the project work in progress documentation, containing reference to the project documentation on Alfresco and has been set up to match the 5G-MEDIA project structure.

Both these services are hosted on ENG cloud infrastructure, and are available at the URL https://production.eng.it/portal/group/prj_5g_media/home, after successful login as shown below in the Figure 7 and Figure 8.

![Figure 7 - Alfresco document repository for 5G-MEDIA](image)

2.2. **Version Control System**

2.2.1. *Description of the tools, comparison and selection among alternatives*

A Version Control System (VCS) helps a team to manage changes to contents over time and to keep track of every modification. This may involve documentation as well as configuration and software code, with the latter becoming critical for every company whose business value is contained in its software assets.

Especially for software code, with developers continuously writing new source code and modifying existing features, keeping track of any change helps preventing concurrent work from conflicting.

The primary features of a VCS are:

- a complete history of changes available for each file over time, with reference to the author and their purpose;
- the possibility to go back to a previous release and check the differences;
- branching, that is the ability to work on different features concurrently and to merge them on the main branch (or the one used to produce an artifact) using a common workflow to manage such branches.

VCSs have deeply changed over time as new features have been provided according to the best practices coming from the developers’ communities; among the most used VCS types there are:
• CVS (Concurrent Version System)\textsuperscript{10} 
• SVN (Subversion) \textsuperscript{11} 
• GIT\textsuperscript{12} 

whose interest is shown in the Figure 8, where Git excels by far over the others.

![Google Trends](image)

**Figure 9 - VCS usage trends (Source: Google Trends [Google6])**

Git main features are described in the official documentation at the URL \url{https://git-scm.com/about/branching-and-merging}, with decentralization being one of the most appreciated by the developers, as it provides source versioning with both local as well as remote repositories even when offline.

Many service providers offer free Git repositories, such as Github\textsuperscript{13}, Gitlab\textsuperscript{14} and Bitbucket\textsuperscript{15}, with slight differences about pricing for private repositories and additional features such as bug tracking, wiki space and other tools for 'social coding'; all have in common the full support to Git protocol but so far Gitlab is the only one to have an open-source version available (Community Edition) that can be used to provide a private repository for the project and a considerable amount of developers.

Git is basically a protocol but does not say anything about the workflow to use in a project to handle concurrent working on the same source code in order to manage different features and issues.

\textsuperscript{10} CVS, \url{https://it.wikipedia.org/wiki/Concurrent_Versions_System}  
\textsuperscript{11} SVN, \url{https://it.wikipedia.org/wiki/Subversion}  
\textsuperscript{12} GIT, \url{https://git-scm.com/}  
\textsuperscript{13} Github, \url{https://github.com/}  
\textsuperscript{14} Gitlab, \url{https://about.gitlab.com/}  
\textsuperscript{15} Bitbucket, \url{https://bitbucket.org/}
The choice of a common *git workflow* is crucial for the teams to work concurrently. There are many of them suggested by the developers’ community and a well-known comparison reported by Atlassian [Atlassian11]; this highlights the difficulties of the adoption of a classic “git-flow” workflow in CD with a different branch for each feature, plus one for the release to contain each new feature to be tested and finally the master release to contain the production ready version. An example is shown in the Figure 10.

![Figure 10 - Git flow example (source: Gitlab.com)](source: Gitlab.com)

On the contrary, a simpler git workflow such as *GitHub flow* (see Figure 11) suggests to use only feature branches that are merged on the master branch when completed, in order to have the master branch ready for staging. This approach reduces the amount of code not yet

---

committed that is closer to the CD and is more appropriate whenever many releases of the same component have to be supported at the time.

![Diagram](Figure 11 - simple GitHub flow (source: Gitlab))

This workflow still requires tagging and merging when dealing with different release versions and also assumes it is always possible to release some code in production.

It also assumes that it is always possible to release in staging the master branch code, that could be a hard requirement to meet depending on the environment to be used; in this case, a possible small variation is Gitlab flow[18] that uses a production branch to be used each time the code is considered ready to be released together with the master branch used for staging. This could be also obtained restricting the access to master branch and having a developer (or a few of them) have a role “master” over the project and take such decision.

To enable CD and still manage different possible features developed concurrently on their own different branch, the merging could be done using “rebase” command, replaying all the feature commits on the top of the master branch, as shown in the Figure 12.

---


2.2.2. Tool available for the project, configuration and guidelines

Gitlab is the Version Control System service available in 5G-MEDIA; the link to the official documentation is https://tinyurl.com/5gmedia-jiradoc.

It is hosted on ENG cloud infrastructure, the installed version is the Community Edition 9.0.6 and the related documentation is https://production.eng.it/gitlab/help; it is available at the URL https://production.eng.it/gitlab, after successful login, as shown in the Figure 13.

The embedded issue tracking service has been disabled in order to have a unique tool dedicated to that purpose, JIRA, described later in this document.

The creation of new projects on Gitlab must be requested to ENG that will provide master role on such project to the requester that, in turn, will assign developer role to the users involved.

---

19 Simple workflow for Continuous Deployment by Atlassian, https://www.atlassian.com/blog/archives/simple-git-workflow-simple
Each *master* can then choose the desired Git flow, although the “Gitlab flow” is among the most appropriate ones given the considerations above.

### 2.3. Issue Tracking System

#### 2.3.1. Description of the tools, comparison and selection among alternatives

An Issue Tracking System is a tool to manage a list of problems within an organization and follow the progress until they are resolved.

A problem or “issue” can be a feature, a bug or any other request the organization wants to keep track of the progression towards its resolution and know who is responsible for resolving it according to a predefined workflow that may vary depending on the type of issue that is going to be processed.

To facilitate management, issues can be search and grouped by category, priority, author and other customizable criteria.

The offerings about Issue Tracking Systems is huge and continuously changing, even the comparison affects many aspects regarding licensing, customizable workflows and fields.[Issue2]

Among these products, JIRA excels like few others, has been ranked as the most popular tool for issue tracking and one of the first for project management [Issue3], and has been continuously supported since 2002 with plenty of plugins available from the community.

#### 2.3.2. Tool available for the project, configuration and guidelines

JIRA is the issue tracking service available in 5G-MEDIA, the link to the official documentation is [https://tinyurl.com/5gmedia-jira](https://tinyurl.com/5gmedia-jira).

It is hosted on ENG cloud infrastructure, the installed version is 6.4.13 and it has been configured with the “Jira Agile SCRUM” workflow as it is one of the default configuration available and it can be easily used for software development, configuration management and bug tracking. It supports three kinds of issues: epics and stories (for Agile SCRUM project management) and bugs; they can be tagged using multiple “labels” for better classification.

About the epics and stories, the suggested workflow is the official Agile SCRUM; about the bugs, given the characteristics of 5G-MEDIA project, it has set up to the typical easy AGILE activity/bug workflow shown in the Figure 14 with the opened (“To do”), working (“in progress”) and completed status (“done”).
The service is hosted on ENG cloud infrastructure and is available at the URL [https://production.eng.it/jira](https://production.eng.it/jira), after successful login, as shown in the Figure 15.

The JIRA service is configured to host different projects, one for each work package (WP) in 5G-MEDIA related to software development.

This is to provide both a project management tool and a bug tracking tool for each WP; each WP leader is granted access as administrator on such WP project in order to define project
management issues (epic, stories, activities) and software issues (bugs), together with the possibility to define a set of Jira “components” to better classify such issues according to the project needs, as shown in the Figure 16.

![Figure 16 - Jira component management to classify issues](image)

### 2.4. Automation Server

#### 2.4.1. Description of the tools, comparison and selection among alternatives

An automation server helps to automate the non-human part of software development process, with hundreds of plugins to support building, deploying and automating any project for the continuous integration.

The main purpose is to integrate every change available on the code repository such that no errors can arise without developers noticing them and correcting them immediately, either through exhaustive reports or notifications.

Just like other continuous integration products, the offerings about automation server is huge and a graphical comparison from G2 Crowd Grid\(^\text{20}\) shows Jenkins among those with high performance and better enterprise user ratings (see Figure 17).

---

\(^{20}\) G2 Crowd Grid statistics, [https://www.g2crowd.com/static/g2_grid_scores](https://www.g2crowd.com/static/g2_grid_scores)
Moreover, Jenkins is open source, provides full support to many builders (e.g. Ant, Maven, Ruby, Python, etc.), the integration with different IDEs (Eclipse, IntelliJ Idea, NetBeans) and with JIRA, and a web interface for an easier configuration as shown in the Figure 18.
Jenkins comes with a set of default plugins (e.g. to support Java programming language\textsuperscript{21} and Maven\textsuperscript{22} build tool); it may be extended with other plugins on request, such as Gradle\textsuperscript{23}.

### 2.4.2. Tool available for the project, configuration and guidelines

Jenkins is the automation server for the continuous integration available in 5G-MEDIA, the link to the official documentation is [https://jenkins.io/doc/](https://jenkins.io/doc/) (for version 2.7.2).

It is hosted on ENG cloud infrastructure, the installed version is 2.73.1 and the service is available at the URL [https://production.eng.it/jenkins](https://production.eng.it/jenkins), after successful login, as shown in the Figure 19.

---

\textsuperscript{21} Java programming language official website, [https://java.com/en](https://java.com/en)

\textsuperscript{22} Maven tool official website, [https://maven.apache.org/](https://maven.apache.org/)

\textsuperscript{23} Gradle building tool official website, [https://gradle.org/](https://gradle.org/)
In 5G-MEDIA, Jenkins is configured to create new builds on demand or automatically every 15 minutes and produce a detailed report about the status and possible errors found. It also supports the execution of post build tasks to enable the integration tests of one or more artifact over a preconfigured environment (e.g. a container, as described later in this section).

The creation of a new job should be requested to ENG that will provide authorization to the requester. Details about the specific configuration needed to run Jenkins tasks in ENG environment are provided in the 5G-MEDIA project technical guide [5G-MEDIA20].

Jenkins has been configured to support default programming languages (Java); support to others, through the installation of the appropriate plugin, should be requested to ENG.

2.5. Binary Repository Manager

2.5.1. Description of the tools, comparison and selection among alternatives

The advantages of a Binary Repository Manager in the continuous integration pipeline are unarguable, as it provides a consistent and reliable access to artifacts integrated with the build environment.

Moreover, the spread of tools like Maven for building and automatic dependency management has increased the importance of using a Binary Repository Manager as a central shared repository for the organization artifacts, saving time and bandwidth, avoiding useless duplications and version mismatches on the developers’ computers.

An extensive comparison among the main Binary Repository Managers is shown in Figure 20, showing a few not recent open source options such as Archiva 24, Artifactory 25 and Nexus 26 that are mostly similar, although Nexus is among one of the most appreciated among developers according to a recent ranking (Aug. 2017).

24 Archiva repository, https://archiva.apache.org/index.cgi
26 Nexus repository, https://www.sonatype.com/nexus-repository-sonatype
2.5.2. Configuration and guidelines for the project

Nexus is the repository manager available for 5G-MEDIA, the link to the official documentation (for version 2.14.0) is [https://tinyurl.com/5gmedia-nexus](https://tinyurl.com/5gmedia-nexus).

It is hosted on ENG cloud infrastructure, the installed version is 2.10.0 and it has been configured to host production-ready and snapshot releases of the software artifacts, both as private repository accessible after successful login.

The URLs to reference the artifacts in the building scripts (e.g. Maven or Gradle) are
[https://production.eng.it/nexus/content/repositories/5G-MEDIA](https://production.eng.it/nexus/content/repositories/5G-MEDIA) (production) and
[https://production.eng.it/nexus/content/repositories/5G-MEDIASnapshots](https://production.eng.it/nexus/content/repositories/5G-MEDIASnapshots) (snapshot).

The service is available at the URL [https://production.eng.it/nexus](https://production.eng.it/nexus), after successful login, as shown in Figure 21.
Artifacts are loaded using Jenkins tasks that are configured to use such repositories through a specific section of the build file (e.g. Maven pom.xml), as described in the technical guide [5G-MEDIA20].

2.6. Code Quality Service

2.6.1. Description of the tools, comparison and selection among alternatives

Tools for the continuous inspection of code quality perform automatic static analysis of source code to check coding standards usage, find source code duplication, detect bugs and security vulnerabilities supporting many different programming languages.

These tools aim at producing a cleaner code with a common code style, so enhancing maintainability.

The offering of such tools is impressive, and a long list is available at the URL https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis in the section “Multi-language”.

SonarQube is an open source project used by more than 80.000 organizations, has a web interface, can detect tricky issues and possible logic errors, and can be included in a CI/CD pipeline along with the build stage using callbacks (“webhooks”) and a comprehensive RestAPI.

2.6.2. Configuration and guidelines for the project

SonarQube is the service for continuous code quality inspection available in 5G-MEDIA, the official documentation is https://docs.sonarqube.org/display/SONAR/Documentation.

It is hosted on ENG cloud infrastructure, the installed version is the Community Edition 6.4; the service is available at the URL https://production.eng.it/sonar, after successful login, as shown in Figure 22.

A report about the source code quality is available only after the first successful execution of the Jenkins task starting from the Gitlab project containing such code.

2.7. Containerization tools

2.7.1. Description of the tools, comparison and selection among alternatives

Containerization is a technology to isolate an application and its dependencies into a self-contained element, similar to VM. Nevertheless, containers adopt a different architectural approach, as they run on the same machine (either bare metal or virtual) using *namespaces* to isolate computing resources assigned to each specific one and *cgroups* to limit the resources usage compared to those assigned to other containers.

Each container is composed by a stack of read-only layers (*images*) stacked on top of each other and holding only the differences with the layer immediately below; each container has on top the only writeable layer that customize it and that is the only one deleted when the container is removed.

This approach promotes reuse of images and lower overhead as it does not require a full virtualized system as VMs do, just a preconfigured environment (e.g. a running “Docker Engine”), and allows to emulate “any environment” just as requested in the *deployment pipeline* definition from Martin Fowler [Fowler14].

But differently from simple VMs, as even the writeable layer can be stored and made immutable (as a new *image*), this promotes version control and more manageable changes in a CI and CD environment. Such *images* are stored on a repository called *registry* and made available to the project.

The fastest growing containerization tool is Docker but some alternatives are available such as CoreOs with rkt28 or LXD29, but given the impressive growth rate and the community support (according to DockerCon 2016 shown in Figure 23 and to an independent report from

---

28 CoreOS with rkt website, [https://coreos.com/rkt/](https://coreos.com/rkt/)
29 LXD website, [http://www.ubuntu.com/cloud/lxd](http://www.ubuntu.com/cloud/lxd)
Datadog shown in Figure 24) Docker is considered among the most suitable containerization tools for production environment.

Given the strong investments and community support, the list of Docker resources available is really compelling with a thorough table updated regularly by a group of independent developers [Docker5]; this also provides the main categories of tools that may be used together with Docker such as:
• build (e.g. docker make\textsuperscript{30}, packer\textsuperscript{31}, docker-maven-plugin\textsuperscript{32}, etc.) for Docker image building and deploying;
• registry (e.g. DockerHub\textsuperscript{33}, Docker Registry\textsuperscript{34}, etc.) for secure storing of Docker images;
• orchestration (e.g. Docker Swarm\textsuperscript{35}, Kubernetes\textsuperscript{36}, Openshift\textsuperscript{37}, etc.) for managing workloads for nodes and balancing container deployments across a cluster.

The management of containers can benefit from specific container-orchestration tools that simplifies scheduling, cluster management and service discovery. A recent survey from New Stack [NewStack17] on this subject shows that Kubernetes is still the orchestration product with higher expectation usage, although Ansible results as a possible alternative even if it is more general purpose.

2.7.2. Configuration and guidelines for the project

Docker is the tool for containerization used in 5G-MEDIA to efficiently create virtual environments for integration tests and staging; the link to the official documentation is https://docs.docker.com/ (version 17.03).

The registry for Docker images may be deployed on the project private cloud (e.g. on ENG infrastructure) or be the one provided for free by DockerHub at the URL https://hub.docker.com/ that makes one private repository available even with a “free plan”\textsuperscript{38}. The choice is dependent on the network latency requirements of the project that are still to be evaluated, so a registry service is not yet provided, although its setup is claimed to be straightforward\textsuperscript{39}.

The container-orchestration features that may be needed in the project are often offered transparently by the NFV MANO framework [MANO12] that is going to be used in 5G-MEDIA

\textsuperscript{30} docker-make, Github, https://github.com/CtripCloud/docker-make
\textsuperscript{31} packer tool for Docker, https://www.packer.io/docs/builders/docker.html
\textsuperscript{32} docker Maven plugin by Fabric8.io, https://github.com/fabric8io/docker-maven-plugin
\textsuperscript{33} dockerhub website, https://hub.docker.com/
\textsuperscript{34} Docker registry tool, https://github.com/docker/distribution
\textsuperscript{35} Docker Swarm website, https://docs.docker.com/engine/swarm/
\textsuperscript{36} Google Kubernetes website, http://kubernetes.io
\textsuperscript{37} OpenShift website, https://www.openshift.com/
\textsuperscript{38} Docker billing plans, https://hub.docker.com/billing-plans/
\textsuperscript{39} Docker registry installation, https://docs.docker.com/registry/deploying/
and that provide specific orchestration over VNFs (e.g. SONATA 40 or OSM41). As this framework is still to be evaluated, generic container-orchestration tools will not be described in this document.

2.8. Configuration management tools

2.8.1. Description of the tools, comparison and selection among alternatives

The advantages of using a configuration management tool to promote automatic operations over different environments are unarguable:

- enforcement, to ensure that a machine is configured to the desired state and avoid configuration drift42;
- configuration files that can be human readable and can be shared through a version control system and keep track of the change control processes;
- abstraction over the scripting languages supported by different operating systems through the usage of a unique (high level) programming language.

More in general, the managing and provisioning of IT infrastructure resources through code rather than using a manual process is called Infrastructure as Code43 (IaC). It promotes the usage of scripts to automate processes (just like CD does) dealing with infrastructure management, using a higher level or descriptive programming languages in order to simplify scripts development and maintenance.

A comparison of open source change management tools is provided by Wikipedia [Wikipedia16] and list of the most popular ones is provided by a recent report from the Linux Foundation [Linux18], resulting in Chef 44, Puppet 45, Ansible 46 and Salt 47.

As 5G-MEDIA project is still in the early stages and specific requirements about operation have not been elicitated yet, it is difficult to define a rationale for the selection, though a first list of differences that can be considered are:

- procedural style (Chef, Ansible), with a description of the steps to execute in order to reach an end state, or declarative style (Salt, Puppet) with a description of the desired end state without specifying the steps to do;
- client/server architecture (Chef, Salt, Puppet) or client only (Ansible) architecture.

40 Sonata website, https://5g-ppp.eu/sonata/
41 OSM, https://osm.etsi.org/
42 Configuration Drift, http://www.continuitysoftware.com/blog/what-is-configuration-drift/
44 Chef, https://www.chef.io/
45 Puppet, https://puppet.com/
46 Ansible, https://www.ansible.com/
47 Saltstack, http://saltstack.com/
A client only architecture such as the one in Ansible simplifies the management of new host/VM as it does not require agents to be installed; on the other hand, the adoption of a tool with greater support due to a larger market share such as Puppet may result in more reliable solution. So, a first choice could be considering both Ansible and Puppet as candidate solutions although taking into consideration the possible changes due to specific needs that may rise in the next future.

2.8.2. Configuration and guidelines for the project

There are no specific guidelines for the usage of change management tools in 5G-MEDIA as they are strictly depending on the operation stage. So far, Puppet and Ansible look like the most appropriate tools for change management even though other could be considered over the project time life.

2.9. Serverless computing: OpenWhisk

2.9.1. Description of the tool

Apache Incubator OpenWhisk (OW) is a serverless open source platform that executes functions (termed “actions”) in response to happening of events (“triggers”). Triggers and actions are joined together by “rules”. Also, actions can automatically execute in pre-defined sequences with output of one action being an input for the next one. A generalization of sequences into more general action composition became available recently\(^48\). Figure 25 depicts OW programming model while Figure 26 shows OW system architecture with the main components. In what follows, we will walk through the architecture components assuming that a user wishes to invoke an action.

1. OpenWhisk’s user-facing API is HTTP based and follows a RESTFul design. An entry point into the system is through NGINX\(^49\), which is an HTTP and reverse proxy server that exports the OpenWhisk REST API. NGINX terminates SSL and forwards the HTTP call to the next component, which is called Controller.

2. Controller is a Scala-based implementation of the REST API used for CRUD requests to create, read, update, and delete OpenWhisk assets including actions, rules, triggers, packages, etc. The Controller disambiguates the REST call based on the HTTP method

\(^{48}\) Rodric Rabbah, “Composing Functions into Applications: the serverless way”, https://medium.com/openwhisk/composing-functions-into-applications-70d3200d0fac

\(^{49}\) Nginx, https://nginx.org/en/
(in case of an action invocation it will be POST) and consults CouchDB\(^{50}\) component for Authentication and Authorization.

3. Metadata about the user and its asset is fetched from the CouchDB and user’s credentials to perform the operation are evaluated.

4. After verifying the user’s privileges to perform the operation and making sure that the user is allowed to invoke the action, the Controller loads a required action (the name is passed as a parameter of the REST call) from the database (i.e., CouchDB). The action record includes the code to run and the metadata that includes limits on resource consumption (e.g., CPU and RAM) and default parameter values to use with the action.

5. The Load Balancer component of the Controller checks with Consul\(^{51}\), which Invokers are available to run the action and selects one of them for this action trying to simultaneously level load across the controllers and increase affinity of the action execution to increase chances that a warm container is available to execute this action at a selected invoker.

6. The Controller does not send the action for invocation to the selected Invoker directly, but publishes it on a Kafka\(^{52}\) topic to which the Invoker is subscribed. The action invocation request enters a line and will be picked up by the Invoker selected by the Controller. Once the action invocation request is published on Kafka, the Controller returns an invocation ID to the client.

7. The Invoker consumes the action invocation request from the Kafka topic. The Invoker is a Scala-based component. It prepares an isolated execution environment, a Docker container, for the action. In case of standard actions (i.e., actions geared towards a specific programming language), a template Docker image exists and Invoker injects the action code into this container image and runs it using Docker REST API (/init and /run).

8. The results of action invocation (if any) are stored in CouchDB as an activation record addressable by the invocation ID. The invocation record also contains a log record produced by Docker. The user can query the invocation record at any point using the invocation ID.

OW can be provisioned on top of any IaaS or within a Data Center infrastructure on VM or bare metal.

OW actions are of two types: standard and blackbox ones. The standard actions relate to specific language binding. There is a standard Docker container image for each standard action corresponding to a specific language environment. A standard action’s metadata and code are stored in the central action repository. When an action is invoked, the code is injected into the

---

\(^{50}\) CouchDB, [http://couchdb.apache.org/](http://couchdb.apache.org/)

\(^{51}\) Consul.io, [https://www.consul.io/](https://www.consul.io/)

\(^{52}\) Kafka, [https://kafka.apache.org/](https://kafka.apache.org/)
image and contained is run. For blackbox actions, custom images are maintained and executed on demand.

OW provides a number of resource limits, which are configurable parameters defining action’s maximal execution time, maximal memory limits, maximal parameter size, concurrency control level, etc.

Figure 25 - OpenWhisk Programming Mode

---

2.9.2. Configuration and guidelines for the project

The 5G-MEDIA project will manage multiple cloud environments: one with a more coordination role ("central cloud") and the others closer to the production/consumption venues ("edge cloud"). For this reason, an instance of OW will be installed on 5G-MEDIA central cloud and additional instances of OW also will be installed at the edges. We refer to this setting as a federated model. In the federated model, there is an OW leader (the one in the central cloud) and a number of followers (at the edges). The OW leader serves as a single point of truth with respect to all OW assets. The follower OW instances automatically keep their assets in sync with the leader according to the eventual consistency model. Such, if an action is being updated at the leader, the update will automatically propagate to the followers. Currently, the federated model is not supported out of the box. In the 5G-MEDIA project IBM will enable the federated model for OW to allow for local action invocations to reduce latency and bandwidth in the 5G-MEDIA use cases.

A specialized reduced resources footprint OW distro will be provided by IBM for the edge environments that might have resource constraints. For the unconstrained environments, a regular OW can be installed.

Initially, the central cloud (i.e., the leader) controller and invokers are going to be installed as Linux VMs each with 8 vCpus, 16GB ram and 50GB of disk (estimated). These VMs are going to be deployed on the same network segment so that OW components have low latency.

---

54 OpenWhisk architecture, [https://thenewstack.io/behind-scenes-apache-openwhisk-serverless-platform/](https://thenewstack.io/behind-scenes-apache-openwhisk-serverless-platform/)
As shown in Figure 26, OW is based on a number of applications and tools, which get automatically installed (such as Gradle, Docker, Ngnx, Kafka, Zookeeper, and Consul). However, Ansible and CouchDB are needed to be installed prior to installing OW.

There is currently no special requirement for configuring OW to use a local Docker Registry, since FaaS execution path does not interact with the registry but with OW local catalog.

3. High-level CI/CD pipeline for the project

A CI/CD pipeline represents the connection among the various tools (such as those described in the previous chapter) to provide the automations and feedbacks the developers expect when adopting a DevOps approach.

All these tools will be subject to periodic update in order to fix possible bugs and support new available features in case of the most “recent” tools.

According to the application specific needs general guidelines about CI and CD from Martin Fowler [Fowler14], such pipeline typically contains a VCS repository (Gitlab) to store source and configuration code, extracted by an automation engine (Jenkins) periodically or automatically after each change on the VCS, built retrieving dependencies by a binary repository (Nexus).

During the build phase, unit tests are executed automatically and errors reported in the automation engine console, in order to stop the pipeline execution if any of them occurs.

Integration tests may rely on containerization to run on the desired environment based on preconfigured test image containers (Docker) retrieved by an image repository (Docker registry); such tests can be executed as a Jenkins post build action and be part of the integration steps handled by this automation server, that provides a detailed report about the whole process.

Once tests are completed successfully, the artifacts are then packaged into domain specific applications (such as a VNF) through configuration management scripts stored on the VCS (e.g. with Ansible, Puppet, etc.) prepared by the developers according to the application specifications (e.g. as a zip file containing binaries and a configuration file), and finally made available to the developers for staging.

Even the staging may rely on containers (Docker) to reproduce an environment for performance and quality acceptance testing before the deployment to production.

The staging environment may consist in the instantiation of a container image (Docker) retrieved by an image repository (Docker registry) representing the desired environment configuration and may already have application specific tools for testing (e.g. emulators).

The staging environment may be also changed with new tools and saved as a new image on the repository (Docker registry) for future testing, following the DevOps general principles.

All along this process, an issue tracker (JIRA) is available to developers to keep track of any error, software code quality improvement requested or even to track the progress of an activity at development time.
The list of the tools selected so far and their responsibilities in the 5G-MEDIA project is reported in Table 1.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Responsibilities in the project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alfresco + Wiki</td>
<td>Document management and collaborative editing tools</td>
</tr>
<tr>
<td>JIRA</td>
<td>Bug tracking and project management tool</td>
</tr>
<tr>
<td>Gitlab</td>
<td>Version Control System for source code and configuration files</td>
</tr>
<tr>
<td>Jenkins</td>
<td>Build automation manager</td>
</tr>
<tr>
<td>Nexus</td>
<td>Binary repository</td>
</tr>
<tr>
<td>SonarQube</td>
<td>Source Code Quality management</td>
</tr>
<tr>
<td>Docker</td>
<td>Containerization tool</td>
</tr>
<tr>
<td>OpenWhisk</td>
<td>Serverless computing platform</td>
</tr>
<tr>
<td>Ansible/Puppet</td>
<td>Configuration Management tools</td>
</tr>
</tbody>
</table>

*Table 1 - list of the selected tools for the 5G-MEDIA project*

An overall picture of a high-level CI/CD pipeline based on Docker is shown in the Figure 27.

*Figure 27 - high-level CI/CD pipeline based on Docker for 5G-MEDIA project*

The pipeline shown above is specific for containerization technologies (apart from Docker).

For production environments, plain VMs or physical hosts may require the instantiation of a new operating system image, the creation and loading of the necessary services using an application specific *integration engine*. 
In case of VMs instead of containers, such integration engine may interface a Virtual Infrastructure Manager (VIM) (e.g. OpenStack\textsuperscript{55}) to instantiate the required infrastructure according to the requirements of the developers’ functions and that may involve VMs as well as virtual network devices. In this case, to support multiple VIMs at once, it may be helpful to adopt a tool that provides an abstract layer on different VIMs to simplify maintenance of such integration scripts; recent, open source tools such as Spinnaker\textsuperscript{56} provide such feature and could be useful in case multiple VIMs support is going to be provided in the 5G-MEDIA project.

As the different possibilities for production environment, the CI/CD pipeline in 5G-MEDIA project will span up to the testing stage, as the container approach provides a more efficient allocation of resources, is easier to maintain and provides all the benefits described before in previous section.

Nevertheless, in production environment tools like Ansible or Puppet should reduce the deployment time thanks to the IaC approach. Apart from the specific configurations needed, a possible approach based on the reuse of containers as part of the production environment will be evaluated (e.g. the building of VM ISOs with a “Docker Engine” configured to run the Docker image used for testing).

This approach will be also evaluated in 5G-MEDIA because of its similarities with SONATA project\textsuperscript{57} that shares the same application domain. SONATA relies on containers for the CI/CD and provides specific tools for testing/emulation (such as son-emu\textsuperscript{58}, based on Docker) that are most probably going to be used in 5G-MEDIA. And in SONATA, “the Docker containers can also be included and executed within VM images to run them on the service platform” (see the “VNFs” section in the examples documentation [Container21]).

Given the early stage of the project, CI/CD in 5G-MEDIA will mostly focus on the building and deployment in test environment based on containers; nevertheless, the deployment in production approach will be evaluated and reported in the Wiki of the project as a result of the task “T2.3 5G-MEDIA Operation Support Tools”.

With the serverless application model (e.g. OpenWhisk), on the contrary, the infrastructure details are completely hidden to the developers and so are, as a consequence, most of the components of the traditional CI/CD pipeline shown before. The serverless platform usually\textsuperscript{59} provides its own API and tools to automate the building and deployment of new functions, so the CI/CD pipeline reduces to the integration of such tools using custom made scripts to be

\textsuperscript{55} OpenStack virtual infrastructure manager, \url{https://www.openstack.org/}

\textsuperscript{56} Spinnaker multi-cloud continuous delivery platform, \url{https://www.spinnaker.io/}

\textsuperscript{57} SONATA NFV project, \url{http://www.sonata-nfv.eu/content/documentation}

\textsuperscript{58} SONATA emulation tool for testing, \url{https://github.com/sonata-nfv/son-emu}

\textsuperscript{59} CI/CD pipeline for a serverless application in AWS, \url{https://github.com/eeg3/serverless-pipeline}
deployed on Gitlab and triggered automatically using an external scheduler or through an administration console from the same platform.

Apart from the specific serverless architecture capabilities that may vary, some tools (such as “serverless.com”\(^{60}\), “back&”\(^{61}\), and “spotinst”\(^{62}\)) provide an abstraction over the different existing serverless architecture (e.g. OpenWhisk, AWS, Google Cloud Platform, etc.) and have a CLI to build, deploy, auto scale, event-driven functions according to the FaaS paradigm; again, with such frameworks, the entire CI/CD pipeline collapses to a set of scripts that may rely themselves on a source code repository for versioning, on such frameworks for building and deploying and triggered automatically using an external scheduler.

**Conclusions**

This document has described the general tools to support CI/CD in 5G-MEDIA project: as it is still in the early stages, it may be necessary to provide more technical details about the provided tools and even update such list with different ones according to the project needs.

The main changes and additions about the tools actually used will be reported in the Wiki of the project as a result of the task “T2.3 5G-MEDIA Operation Support Tools”.

The configuration and the technical details about the tools have been released on a separated technical guide on the Wiki [5G-MEDIA20] and will be updated along the project time life to be a common reference for all the partners.

---

\(^{60}\) serverless tool, [https://serverless.com/](https://serverless.com/)

\(^{61}\) back& tool, [https://www.backand.com/](https://www.backand.com/)

\(^{62}\) spotinst tool, [https://spotinst.com/products/spotinst-functions/](https://spotinst.com/products/spotinst-functions/)
References


[Container21] the usage of containers in VMs to be run on the SONATA Service Platform, “VNFs (vnfs)” section in examples documentation, https://github.com/sonata-nfv/son-examples