dRural

D4.2 Deployment Planning

WP4 Task 4.2

Authors: IDI EIKON

Reviewers: Roberto Giménez (EUROB), Alejandro García (EMERGYA), Myriam Martin (TICBIOMED)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

Technical references

GA number	GA 101017304
Project Acronym	dRural
Project Title	The service marketplace for European Digital rural areas
Project Coordinator	Myriam Martín TICBIOMED (TBM) Myriam.martin@ticbiomed.net
Project Duration	January 2021 – June 2024 (42 months)

Deliverable No.	D4.2
Title	Deployment Planning
Dissemination level*1	Public
Work Package	WP4 – Solution set-up in each Demonstrator
Task	T4.2 – Local Demonstrator Deployment Planning
Lead beneficiary	6 EIKON
Contributing beneficiary/ies	13 EUROB CREATIVE
Due date of deliverable	M12 2021
Actual submission date	24/01/2022

Abstract

The primary purpose of this deliverable is the use of the Deployment Plan, as a common plan for the pilot partners to manage the pilot execution, but also the WP4 coordinator to monitor the pilot progress.

(*) PU = Public; PP = Restricted to other programme participants (including the Commission Services); RE = Restricted to a group specified by the consortium (including the Commission Services); CO = Confidential, only for members of the consortium (including the Commission Services)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

Disclaimer

v	Date	Beneficiary	Author
01	28.06.2021	EIKON	Josefina Farinós
02	31/08/2021	EIKON	Miguel Alborg (Sr.), Josefina Farinós, Miguel Alborg (Jr), Gabriel Pérez
03	14/10/2021	EUROB	Roberto Giménez, Daniel García
04		TICBIOMED	Myriam Martin
05		EMERGYA	Alejandro García
06	30/11/2021	EIKON	Miguel Alborg (Jr)
10	21/12/2021	EUROB	Roberto Giménez, Daniel García
11	22/12/2021	EIKON	Miguel Alborg (Jr)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

Table of contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

List of Tables

List of Figures

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

Executive Summary

This document provides the detail of the Deliverable D4.2, which is primarily linked to T4.2 Deployment Planning.

The deployment will be based on the **dRural** platform package for Amazon Cloud facilitated by WP3.

This document provides the calendar planning of the tasks to be done for installing a **dRural** environment to both project partners and Regions joining the project under WP5 Open Calls and limiting, despite, the different strategies, the variance in success.

Software deployment procedure will include the steps, processes, and activities required to make a software system or update available to its intended users. Deployment is the mechanism through which applications, modules, updates, and patches are delivered from developers to users.

dRural deployment installation will offer three main alternatives:

- Amazon Web Services (AWS) Cloud
- Microsoft Azure Cloud
- On-premises IT infrastructure

Deployment plan will also include the following elements to build a detailed calendar of actions to undertake:

- Pilot Site Profiling Summary
- Training and Deployment-related materials
- Support and Bug Reporting strategies
- Risk identification and mitigation measures related to deployment operations
- Deployment evaluation

Along D4.2 sections we will outline how all these concepts are interrelated to set up **dRural** Deployment Plan.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

1 Planning a Pilot Implementation

A deployment plan is to define the goals, scope, roles, and responsibilities of key stakeholders, architecture, implementation, and testing. The plan is to ensure a smooth transition to a new product or service and address all possible contingencies to quickly troubleshoot and solve any issue that may occur during the deployment process. The deployment plan also defines a training schedule for those involved in the running of the system.

In **dRural** pilot implementation has to be aligned with customer-specific requirements and will be localised by language and by currency and beyond this by using customized "skins" for end user interfaces for the promoters' organization.

The idea is to demonstrate feasibility by focusing on solving key use cases without ignoring real-world constraints. The aim is to identify and address any issues that may threaten the deployment.

In **dRural** pilot results will be iteratively reviewed since the platform goes through different versions and demonstrators will refine their uses cases based on observations

1.1 Purpose

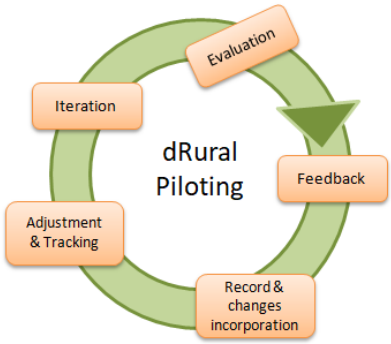
The primary purpose of this deliverable is the use of the Deployment Plan, as a common plan for the pilot partners to manage the pilot execution, but also the WP4 coordinator to monitor the pilot progress.

Deployment refers to moving an object to a place where some action can be performed on it. In the case of **dRural**, deployment means making the **dRural** platform (different releases), ready to be run in a production environment.

Deployment is in charge of doing the instantiation of the four regional demonstrators and releasing it to those partners, ensuring a smooth customer experience. Partners' demonstrators will run and check it, coming back with any potential issue.

In **dRural** development partners follow agile strategies, thus, once the system is deployed, and demonstrators start using the developed and instantiated system, will enter in a continuous iteration cycle feeding WP3.

Figure 1: dRural Iterative Feedback cycle



Feedback – Record & changes incorporation – Adjustments and Tracking – Iteration – Evaluation/Assessment ..... Feedback

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

1.2 Organisation

Beyond connectivity, WP4 activities will evolve around the planning, quality assurance and deployment of the piloting phase(s) for the **dRural** service. Task 4.2 Deployment Planning is the one undertaken in the actual document: deliverable D.4.2, although as scheduled in the DoA, the planning is dependent of the release of the platform versions, coming first a MVP (minimum viable platform), that will be followed by more complex ones, complementing the first release.

dRural pilots are planned to validate the impact, functional and non-functional requirements, mainly usability and interactivity of the services implemented in each of the pilot sites as defined in the Use Cases (WP3), in the four testing sites, thus reaching conclusions for the subsequent uptake of the service.

This WP has the responsibilities to:

- Deploy the service to its running environments
- Validate and Evaluate the technical Service aspects using proper questionnaires and indicators per pilot case and measure the impact.
- Ensure the training for all (T4.5) - as agreed training should have 2 phases: the training of IDI EIKON (training of trainers) and the training of the rest

This section describes in a summary way, how demonstrators are organised, who the stakeholders are, their needs, challenges and their responsibilities. It also provides the dependencies with the external platforms and service providers, required to ensure successful cutover to the new service.

The “Bug” and Evaluation templates to feed WP3 for technical improvements for the next releases are also provided. Risks and risk mitigation strategies to implement are also considered.

During **dRural** project lifetime, demonstrator partners’ main activities are to focus on continuous improvement of the platform to keep it useful and easy to adopt, as well as minimizing its complexity as much as possible. The needs emerged in the deployment operation are to be reported to WP3 and, when possible, handled properly in upcoming new releases of the platform.

At each of the demonstrators' sites it will be carried out a continuous evaluation and collection and analysis of results, in terms of technical merits and functional performance. The main analysis criteria will be the degree to which the technical requirements defined in WP3 - T3.1 are satisfied, as well as other technical key performance indicators such as system stability, reliability, usability, security, etc.

These Deployment Evaluation reports will allow us to analyze the deployment process results, what are the needed improvements, what is the end-users' view on the provided services through **dRural** Platform, and so on.

An analysis on the collected data will be carried out after each major release and its results will be passed to WP3 for technical improvements.

The lessons learnt in the deployment will be used as guide for replication of the pilot sites to other mirror regions, interested to deploy **dRural** solution. This promotion will help project consortium to increase the chances of private or public-private alliances at regional level and into a number of local markets and overall wide deployment of the solution.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 101017304.

2 Platform Releases and Deployment

dRural marketplace platform is foreseen to take place in M18.

Deployment partner will have access to a beta version of this release in M16 in order to perform a set of pre-testing activities and to generate a Minimum Requirements Guide for pilot sites to consider before facing the first-ever installation of a **dRural** Platform.

This Minimum Requirements Guide will be available in M17 and will provide details on which software and hardware infrastructure should be in place before deploying a **dRural** platform release.

In the first release of the platform, dRural marketplace and the core part of dRural metaplatform will be provided.

Following releases (after the first one) are foreseen in:

- M27, Silver Release
- M36, Gold Release
- M42 Final Release

As soon as versions are released, deployment partner will have access to them in order to test and certificate **dRural** platform operation in all deployment modalities, this will be accomplished running automated individual components tests, and a full platform functional evaluation. Once certification runs out a QA test checklist, training materials and deployment guides for new version will be generated for pilots.

Specific Deployment Guides will be produced to be available for all pilot sites to follow 1 month after platforms versions are released (except for Final Release, expected for the end of the project on M42):

Table 1: dRural platform releases and guides

dRural Platform Release	dRural Deployment Guide
Beta Release (only for Deployment Partner to access) – M16	Minimum Requirements Guide – M17
MVP Release – M18	MVP Deployment Guide – M19
Silver Release – M27	MVP Deployment Guide – M28
Gold Release – M36	MVP Deployment Guide – M37
Final Release – M42	MVP Deployment Guide – M42

Deployment guides, containers, helm charts and configuration samples will be deployed via a private repository to all partners.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

3 Deployment and Monitoring

Deployment is really one of the last stages of a development, with a flow (generally speaking), consisting of 5 steps: Planning, Development, Testing, Deploying, and Monitoring.

In **dRural**, Deployment is understood as ensuring **dRural** Platform, in any of its released versions, is running smoothly in a production environment for all four project pilot sites demonstrators.

Having a Deployment Plan ensures that everything is done the same way each time a new installation is set up, any time changes are made and any time new versions for **dRural** platform are released. And in **dRural** this will be especially helpful in the case of those providers joining the project in the Open Calls window (starting on M30).

dRural first time deployment will be a full one, while in successive phases will be incremental.

Once Deployment is successfully completed, services will go live and real end users will start interacting with the **dRural** platform. After that, it is important to recurrently monitor that everything works as expected. No matter the planning put forward, there's always a chance that users encounter issues or perform actions that were not anticipated during the planning and development phases. To help on this, we have designed a bug form template to make easy for demonstrators to give their feed-back and receive support and maintenance

In **dRural**, releases are planned and scheduled for certain dates from the beginning (project Work Plan), and those milestones are known by providers and pilot sites within the **dRural** consortium. Those dates are planned immediately after WP3 releases are made available for the deployer partner.

Deployment work starts after development partner releases and delivers all the artifacts needed for a **dRural** platform installation before a planned deployment milestone. This requires ensuring there is time enough to allow deployment partner to prepare for pilot sites deployment, by:

- Having access to a beta version of the **dRural** platform in order to prepare guides on Minimum Requirements pilot sites must accomplish in terms of infrastructure, connectivity and software/hardware.
- Ensure that development partners train and provide support to deployer in this beta "lab" pre-production environment (training of the trainers).
-
- Ensure that deployer could run and train the pilot regions in adapting and localising its content or service portfolio, where needed, to its peculiarities. This task can be performed regularly as new **dRural** Platform versions are released featuring new localisation capabilities.

First access of the deployer partner to this beta environment is expected to occur around M16.

If no delays happen, in M18 deployer will have access to the **dRural** platform version MVP as released from WP3 by development partner.

At this point, deployer partner will be able to generate the specific Deployment Guide for this MVP version (available in M19), which will be adapted and tailored to pilot sites specific needs. Using this guide, pilot site technical teams should be able to handle the deployment process of this version, even when deployment will be co-ordinated and supported by the deployer partner working jointly with pilot site technical teams.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

Once the initial deployment at all four pilot sites (M20) is completed successfully the monitoring phase will start and pilot sites would start reporting back, using the "bug template" with any potential defect so that developers and deployers can improve or solve any question aroused and look forward to release a next version with new features and improvements.

Finally, deployer partner is in charge of supporting each **dRural** regional partner while they manage the integration of one complex service per each one with an existing platform. Integration in this context is understood as ensuring data/information transfer from one system/platform to another (being this flow unidirectional or bidirectional).

In order to ensure this integration can be managed, pilot site must fulfil a pilot profiling template which may list services to be integrated and relevant information on how to handle this integration: available APIs, Web Services, documentation, security and data protection measures.

Pilot profiling templates should be dully completed at least 1 month before a deployment is foreseen (M20).

This deployment and monitoring phase will be repeated and generate several iterations for all the versions planned to be released during **dRural** project lifetime:

Table 2: dRural deployment planning

dRural Platform Release	dRural Deployment Guide	dRural Deployment Completed
MVP Release – M18	MVP Deployment Guide – M19	M20
Silver Release – M27	MVP Deployment Guide – M28	M29
Gold Release – M36	MVP Deployment Guide – M37	M38
Final Release – M42	MVP Deployment Guide – M42	M42

3.1 Pilot Profiling

As a reminder, in **dRural**, technically speaking, the project deals with two main service components or infrastructures: **dRural metaplatform** and **dRural marketplace**.

Pilot sites will build services on top of these two main components.

As for the metaplatform, the partner contributing the technical solution uses Fiwoo. Fiwoo capabilities are based on Fiware IoT Agents and developer partner, as defined in the DoW, will provide all external **dRural** platform interfaces based on Open Standards.

All Services (simple & complex) will be offered through the **dRural** MARKETPLACE, but complex services will run separately.

For running complex services **dRural** METAPLATFORM is needed and it will be contributed during the project time by free by developer and owner partner, to be used as a standalone platform. Each demonstrator will decide which Fiwoo features to use. If needed, those features can be extended by

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

themselves throughout an SDK they will be facilitated with by the developer partner. Usage of the **dRural** METAPLATFORM will happen after the deployment of **dRural** MVP Marketplace release in M18. It is expected to have **dRural** metaplaform fully ready for **dRural** Silver release in M27.

Development partner will release a **dRural** Beta platform ready for running, using an Amazon EKS Cloud based deployment modality. Although it was initially agreed that all four partners Regions will have the same instance model (Amazon EKS Cloud based), Deployment partner is offering an alternative to those partners that opt for running it:

- Using a Microsoft Azure AKS Cloud based Deployment Modality
- Using an "on premise" Deployment Modality (with the possibility of jumping to the cloud at any moment)

This has been agreed as the technical questionnaire already circulated among pilot sites has shown a disparity of preferences on this deployment modality decision, which may represent a relevant risk to this Deployment Plan.

In order to organize the Deployment calendar at every demonstrator, every pilot site will provide a Pilot Profiling document (see template in 3.1.1), which should be closed by M16, when deployer partner starts to adapt deployment materials according to the Beta release facilitated by the developer partner.

This Pilot Profiling document includes:

- Details on pilot site team involved in the Deployment, including people of the same or related/supportive organisation(s) with different responsibilities in the pilot deployment (e.g technical manager/assistant, administrator, operator).
- A contact list from all people involved in the pilot. Contact details include mainly full name, email, telephone, skype id (or similar) and responsibility role to the pilot deployment.
- Deployment Modality for running **dRural** Platform. Pilots should indicate whether they would use:
    - Amazon EKS Cloud based modality, as originally suggested by developer partner and as originally adapted for **dRural** Beta release
    - Microsoft Azure AKS Cloud based modality, as adapted by deployer partner from the Beta release received from developer partner
    - On-premise Rancher Kubernes Engine (RKE) modality, as adapted by deployer partner from the Beta release received from developer partner
- Relevant integration/interoperability documentation for Complex services to be used within every pilot site, to be provided as soon as they are available
- Main localisation needs as compiled from D4.3

3.1.1 Pilot Profiling Template

Pilot sites are required to complete the following data regarding its pilot activities and service portfolio by M16, so Deployment Plan can roll out as expected:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

Table 3: data requested to regional partners

Key Personnel	• Provide here a bullet list or table with Key Personnel contact details (for those to be actively involved in pilot deployment, management and support activities) • Name 1 + Surname 1 + Role in the pilot + contact details • Name 2 + Surname 2 + Role in the pilot + contact details • …
Needs expressed	Provide here a short list/text (10-12 lines) on main goals expected to be achieved within dRural and, where possible, relate them to potential simple or complex services that will be built. Goal 1 Goal 2 …
Challenges	Provide here a list of the main challenges/topics/keywords that will be addressed using **dRural** pilot sites services to be built. • Topic A • Topic B • Topic C
Deployment Modality	Please mark here which Deployment Modality applies to pilot site: ☐ Amazon Cloud ☐ Azure Cloud ☐ On-Premise
Dependencies with external platforms / service providers	Please provide here, where available, dependencies (in terms of services interoperability) that might be considered when setting up the pilot site service portfolio, especially when discussing Complex Services set up.
Main Localisation needs	Please provide here which might be the main localisation needs (interfaces language, currencies…) and, when possible, link them to pilot site service portfolio, especially when data sources to be used for setting up Complex Services are to be considered.
Notes/Comments:	Provide here relevant comments on issues that may impact **dRural** services set up or running and, in general, to pilot site planned service portfolio.

3.1.2 Pilot Sites Profiling State-of-the-art

Pilot Profiling activities so far (M12) have resulted in the following data being provided by **dRural** 4 pilot sites.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

Demonstrators will have up to M16 to review and complete their Pilot Profiling template.

Table 4: Region Dubrovnik-Neretva (Croatia)

Key Personnel	Name + Surname + Role in the pilot + contact details (as many as key persons involved)
Deployment Modality	☐ Amazon Cloud ☐ Azure Cloud ☐ On-Premise
Dependencies with external platforms / service providers	
Main Localisation needs	
Notes/Comments:	Integrations will happen if possible (both parts are willing to integrate and have the software needed for it or, failing that, develop it by themselves with the SDK to be supplied by developer partner in a further phase). The first version covers only simple services, no integrations.

Table 5: Region Gerderland Midden (Netherlands)

Key Personnel	Name + Surname + Role in the pilot + contact details (as many as key persons involved)
Deployment Modality	☐ Amazon Cloud ☐ Azure Cloud ☐ On-Premise
Dependencies with external platforms / service providers	
Main Localisation needs	
Notes/Comments:	Integrations will happen if possible (both parts are willing to integrate and have the software needed for it or, failing that, develop it by themselves with the SDK to be supplied by developer partner in a further phase). The first version covers only simple services, no integrations.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

Table 6: Region Extremadura (Spain)

Key Personnel	Name + Surname + Role in the pilot + contact details (as many as key persons involved)
Deployment Modality	☐ Amazon Cloud ☐ Azure Cloud ☐ On-Premise
Dependencies with external platforms / service providers	
Main Localisation needs	
Notes/Comments:	Integrations will happen if possible (both parts are willing to integrate and have the software needed for it or, failing that, develop it by themselves with the SDK to be supplied by developer partner in a further phase). The first version covers only simple services, no integrations.

Table 7: Region Jämtland Härjedalen (Sweden)

Key Personnel	Name + Surname + Role in the pilot + contact details (as many as key persons involved)
Deployment Modality	☐ Amazon Cloud ☐ Azure Cloud ☐ On-Premise
Dependencies with external platforms / service providers	
Main Localisation needs	
Notes/Comments:	Integrations will happen if possible (both parts are willing to integrate and have the software needed for it or, failing that, develop it by themselves with the SDK to be supplied by developer partner in a further phase). The first version covers only simple services, no integrations.

3.2 **dRural** Deployment Modalities

dRural Platform will offer its demonstrators three different Deployment Modalities to run **dRural** services, in order to accommodate to preferences and policies applying to every pilot site and looking forward to avoid potential risks when the time to deploy **dRural** released versions arrives.

dRural platform is a composition of multiple components and uses Kubernetes (K8s) for deployment automation and container orchestration. Platform developers will supply deployment scripts and support for Amazon EKS based deployment modalities. However, as some of the pilots prefer to use

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

alternative infrastructure providers, deployment partner will provide deployment guides, scripts and support for two additional deployment modalities. This will allow pilots to choose between three Kubernetes alternatives:

- Amazon Elastic Kubernetes Service (EKS), a managed service allowing Kubernetes cluster management, containers deployment and applications run management capabilities relying on a cloud based infrastructure provided by Amazon Web Services (AWS)
    - Follows the same Deployment Modality used by developer partner for **dRural** platform Beta version
    - Follows the same Deployment Modality recommended by developer partner
    - Follows the same Deployment Modality that has been successfully deployed into production environments in the past, for Fiwoo metaplatform component
- Microsoft Azure Kubernetes Service (AKS). The Microsoft Azure service for Kubernetes deployments on Azure infrastructure.
- On premise Rancher Kubernetes Engine (RKE) modality. An alternative deployment modality allowing pilots using self-hosted or ISP hosted servers infrastructure. RKE it's an open source CNCF-certified Kubernetes implementation allowing easier deployments on Linux. It has no license fees but optional support can be provided by SUSE.

3.2.1 Choosing a Deployment Modality: Making the decision

There is no right or wrong answer to the cloud vs on-premise deployment modality dilemma. Every user/organization is different and has different requirements that will influence the choice of the deployment strategy.

It is wise to lean toward a managed service if:

- The idea of understanding *Kubernetes* sounds terribly arduous
- The on-premise infrastructure is not prepared for *Kubernetes*
- The responsibility for managing a distributed software system that is critical to the success of the business sounds dangerous
- The inconveniences of restrictions imposed by vendor-provided features seem manageable Managed Service Versus Roll Your Own
- You trust your managed service vendor to respond to your needs and be a good business partner

Leaning toward rolling your own *Kubernetes* if:

- The vendor-imposed restrictions make you uneasy
- You have little or no faith in the corporate behemoths that provide cloud compute infrastructure
- You do believe in the power of *Kubernetes* and trust your abilities to build around it.
- You relish the opportunity to leverage this container orchestrator to provide a delightful experience to your developers

dRural first time deployment will be a full one, while in successive phases upcoming deployments will be incremental.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

3.3 Deployment Planning: Steps to be followed by Demonstrators

dRural Deployment Plan has designed the following calendar to ensure a smooth deployment process in all four demonstrator sites:

Figure 2: dRural deployment planning

TASKS	M12	M13	M14	M15	M16	M17	M18	M19	M20	…	M27	M28	M29	…	M36	M37	M38	…	M42
dRural Pilot Profiling					PP														
dRural Version Release					B		MVP				S				G				F
dRural Version Testing						RG		DG				DG				DG			DG
dRural Training(s)							TT		TP				TP				TP		TP
dRural Version Localisation																			
dRural Version Deployment									MVP				S				G		F
Deployment Reporting																			
Deployment Evaluation Report									R				R				R		R
Deployment Support to Pilot Sites																			
Deployment Support to Mirror Regions																			

Gant Diagram Legend:

PP:	Pilot Profiling
B:	Beta version
MVP:	MVP version
S:	Silver version
G:	Gold version
F:	Final version
RG:	Minimum Requirements Guide
DG:	Deployment Guide
TT:	Train-the-trainers sessions
TP:	Train-the-pilots sessions
R:	Post-deployment report

Main relevant sections of the above outlined plan are:

- Ensure Pilot Sites Profiling (PP) is completed by M16, including all the data stated in Section 4.1 of this deliverable. This will set up the basis for deployer partner to adapt deployment guides and materials for pilot site technical teams to use. Recruitment and identification of key personnel involved in the pilot is a key success factor for this and upcoming stages of this deployment plan.
- Ensure deployer partner has access to **dRural** Beta Platform in M16. Access to this version will allow deployer partner to test in advance the basic features of the **dRural** plataform version as well as to generate a Minimum Requirements Guide (RG) for all pilot site to use when setting up their software and hardware infrastructure. This RG guide will allow pilot sites to ensure their infrastructure is ready to deploy **dRural** MVP version in M19-20. Delays in managing properly this milestone may impact in the entire planned deployment calendar.
- Ensure deployer partner has access to **dRural** MVP Platform in M18. This will be first ever available version of **dRural** platform for production environments. Deployer partner must access this version on time to properly generate and adapt Deployment Guides (DG), adapted to every pilot site Deployment Modality, for pilot sites to follow and complete the deployment process by M20.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

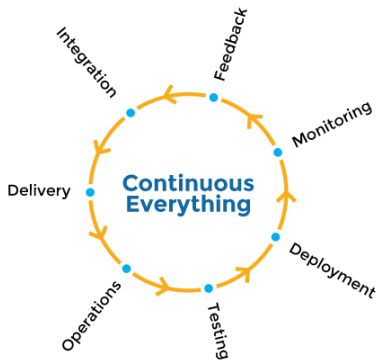
- For upcoming releases of the dRural Platform (M29 – M38 – M42) the Deployer partner will provide new Deployment Guides (DG) on M28, M37 and M42. Those guides will detail how to conduct incremental upgrade processes from a **dRural** version to a new one. In addition, the Deployer partner will review the "first-ever deployment" guide, introducing any relevant aspect that has to be considered as far as Developer partner delivers new versions of the **dRural** Platform.
- In parallel, by M18 deployer must have received the proper training-the-trainers sessions (TT), so DG guides are designed based on this training and training-the-pilots sessions (TP) can happen by M20 at every pilot site.
- In upcoming deployments for newly available **dRural** Platform versions, TP sessions will be repeated based on pilot site needs.
- Pilot site users to be trained in TP session may be, preferably, in-house employees supporting the service and actively engaged in pilot activities. It is recommended to set up a small group per demonstrator (up to 5 persons) able to train in turn other potential users of the organization in case it is needed.
- Relying on Pilot Profiling documents (PP) delivered by pilot sites, Localisation needs will be addressed for every pilot site in between 1 and 2 months a new version is planned to be deployed.
- By M20, pilot site users are expected to be able to start creating, adapting and delivering services in their pilot portfolio to their citizens.
- From M20 onwards, when all pilots sites are expected to be running **dRural** MVP platform version, all pilot sites can start reporting incidences and asking for support using the designed Bug Report template.
- Deployer partner will produce a Deployment Evaluation Report (R) every time a deployment procedure is completed for a **dRural** Platform version. Findings and conclusions of these reports will be shared with WP3 to introduce as many improvements as possible for managing upcoming deployment procedures in the future.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

4 Feed-back to WP3 (bugs reporting)

Demonstrators will report continuously (using the template prepared), the bugs or inconsistencies or missing features they detect, in order to feed WP3 for next platform release.

Figure 3: Continuous Everything cycle



What follows is the instructions and template facilitated to demonstrators for performing such a task:

4.1 Reporting Bugs

Demonstrator's actions to be performed during the running of the pilots:

Characteristics and Techniques include

#1) Having a clearly specified Bug Number: Always assign a unique number to each bug report. This, in turn, will help you to identify it. If you are using any automated bug-reporting tool then this unique number will be generated automatically each time while you report the bug.

Note the number and a brief description of each bug that you reported.

#2) Reproducible: If your bug is not reproducible, then it will never get fixed.

You should clearly mention the steps to reproduce the bug. Do not assume or skip any reproducing step. A bug which is described step by step is easy to reproduce and fix.

#3) Be Specific: Do not write an essay about the problem.

Be Specific and to the point. Try to summarize the problem in minimum words yet in an effective way. Do not combine multiple problems even if they seem to be similar. Write different reports for each problem.

Effective Bug Reporting

Bug reporting is an important aspect within the demonstrators activities. An effective Bug report communicates well with the development team and avoids confusion or miscommunication.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

A good Bug report should be **clear and concise** without any missing key points. Any lack of clarity leads to misunderstanding and slows down the development process as well. Defect writing and reporting is one of the most important but neglected areas in the testing life cycle.

Good writing is very important for filing a bug. The most important point that a tester should keep in mind is **not to use a commanding tone** in the report. This breaks the morale and creates an unhealthy work relationship. Use a suggestive tone.

Don't assume that the developer has done a mistake and hence you can use harsh words. Before reporting, it is equally important to check if the same bug has been reported or not. Although a duplicate bug can be a burden (for developers), in case of doubt, report it.

The important information that a bug report must communicate is **"How?" and "Where?"** The report should clearly answer how the process was performed and where the defect occurred exactly. The reader should easily reproduce the bug and find where the bug is.

Keep in mind that the **objective of writing the Bug report** is to enable the developer to visualize the problem. He/She should clearly understand the defect from the Bug report. Remember to give all the relevant information that the developer is seeking.

Also, bear in mind that a bug report would be preserved for future use and should be well written with the required information. **Use meaningful sentences and simple words** to describe your bugs. Don't use confusing statements that wastes the time of the reviewer.

Report each bug as a separate issue. In case of multiple issues in a single Bug report, you can't close it unless all the issues are resolved.

Hence it is best to **split the issues into separate bugs**. This ensures that each bug can be handled separately. A well-written bug report helps a developer to reproduce the bug at their terminal. This helps them to diagnose the issue as well.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

4.2 Bug Reporting Template

Table 8: Bug Reporting Template

Reporter: **(example)**	Your name and email address Xxxxx Xxxxxx from Los Santos de Maimona, Local Council xxxxx@hotmail.com Extremadura Region (Spain) demonstrator
Version **(example)**	The product/service version if any MVP
Category: **(example)**	These are the major sub-modules of the product/service Marketplace, Fiwoo, Deployment, Backend
Bug Number/id **(example)**	MAI-B-001 (3 letter for Maimona; B for Bug; number order -3 digits-starting from 1
Bug Title	
Priority:	Priority is set from P1 to P5. P1 as "fix the bug with the highest priority" and P5 as "Fix when time permits"
Operating System	Mention all the operating systems where you found the bug. Operating systems like Windows, Linux, Android, Mac OS. Mention the different OS versions also like Windows 10, Windows 11 etc, if applicable
Browser	Edge, Chrome, Firefox... and version, e.g. Firefox 85.0.1 (64 bit)
Description:	A detailed description of the bug **Use the following fields for the description field:** **Reproduce steps**: Clearly, mention the steps to reproduce the bug. **Expected result**: How the application should behave on the above-mentioned steps. **Actual result**: What is the actual result of running the above steps i.e. the bug behaviour
Attachments:	Files or Screenshots
Status:	When you are logging the bug into any bug tracking system then by default the bug status will be 'New'. Later on, the bug goes through various stages like Fixed, Verified, Reopen, Won't Fix, etc. **States:** **New; Assigned; Open; Fixed; Pending Retest; Retest; Reopen; Verified; Closed** **Few more:** Rejected; Duplicate; Deferred; Not a Bug
URL:	The page URL on which the bug occurred

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

Explanation of the different fields' inputs

#1) Bug Number/id

A Bug number or an identification number (like b001) makes bug reporting and referring to a bug much easier. The developer can easily check if a particular bug has been fixed or not. It makes the whole testing and retesting process smoother and easier.

#2) Bug Title

A Bug title is read more often than any other part of the bug report. It should say all about what comes in the bug.

The Bug title should be suggestive enough that the reader can understand it. A clear bug title makes it easy to understand and the reader can know if the bug has been reported earlier or has been fixed.

#3) Priority

Based on the severity of the bug, a priority can be set for it. A bug can be a Blocker, Critical, Major, Minor, Trivial, or a suggestion. A bug priority from P1 to P5 can be given so that the important ones are viewed first.

#4) Environment

The OS and browser configuration is necessary for a clear bug report. It is the best way to communicate how the bug can be reproduced.

Without the exact platform or environment, the application may behave differently and the bug at the tester's end may not replicate on the developer's end. So it is best to mention clearly the environment in which the bug was detected.

#5) Description

Bug description helps the developer to understand the bug. It describes the problem encountered. The poor description will create confusion and waste the time of the developers and the testers as well.

It is necessary to communicate clearly about the effect of the description. It's always helpful to use complete sentences. It is a good practice to describe each problem separately instead of crumbling them altogether. Don't use terms like "I think" or "I believe".

#6) Steps to Reproduce

A good Bug report should clearly mention the steps to reproduce. The steps should include actions that cause the bug. Don't make generic statements. Be specific in the steps to follow.

#7) Expected and Actual Result

A Bug description is incomplete without the Expected and Actual results. It is necessary to outline what is the outcome of the process and what the user should expect. The reader should know what the correct outcome of the process is. Clearly, mention what happened during the process and what was the outcome.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

#8) Screenshot

A picture is worth a thousand words. Take a Screenshot of the instance of failure with proper captioning to highlight the defect. Highlight unexpected error messages with light red colour. This draws attention to the required area.

4.3 The importance of defects

It is important to know about the various states of a defect since the main intention of performing this activity is to check if the product has any issues/errors. The main objective of doing bug reporting is to assure that the product is less prone to defects (no defects is an unrealistic situation).

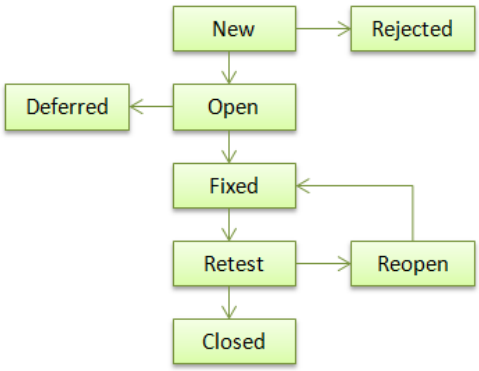
A Defect, in simple terms, is a flaw or an error in an application that is restricting the normal flow of an application by mismatching the expected behavior of an application with the actual one.

It is the responsibility of demonstrators to try thoroughly the application to find as many defects as possible to ensure that a quality product will reach the customer. For reporting properly, it is important to understand about defect life cycle. A Defect life cycle, also known as a Bug life cycle, is a cycle of a defect from which it goes through covering the different states in its entire life. This starts as soon as any new defect is found and comes to an end when a tester closes that defect assuring that it won't get reproduced again.

Defect Workflow

It is easier to understand the actual workflow of a Defect Life Cycle with the help of a simple diagram as shown below.

Figure 4: Defect Workflow



Defect States

- **1) New:** This is the first state of a defect in the Defect Life Cycle. When any new defect is found, it falls in a 'New' state, and validations and testing are performed on this defect in the later stages of the Defect Life Cycle.
- **2) Assigned:** In this stage, a newly created defect is assigned to the development team for working on the defect. This is assigned by the project lead or the manager of the testing team to a developer.
- **3) Open:** Here, the developer starts the process of analyzing the defect and works on fixing it, if required. If the developer feels that the defect is not appropriate then it may get

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

transferred to any of the below four states namely **Duplicate, Deferred, Rejected, or Not a Bug**-based upon the specific reason.

- **4) Fixed:** When the developer finishes the task of fixing a defect by making the required changes then he can mark the status of the defect as 'Fixed'.
- **5) Pending Retest:** After fixing the defect, the developer assigns the defect to the tester for retesting the defect at their end, and till the tester works on retesting the defect, the state of the defect remains in 'Pending Retest'.
- **6) Retest:** At this point, the tester starts the task of working on the retesting of the defect to verify if the defect is fixed accurately by the developer as per the requirements or not.
- **7) Reopen:** If any issue persists in the defect then it will be assigned to the developer again for testing and the status of the defect gets changed to 'Reopen'.
- **8) Verified:** If the tester does not find any issue in the defect after being assigned to the developer for retesting and he feels that if the defect has been fixed accurately then the status of the defect gets assigned to 'Verified'.
- **9) Closed:** When the defect does not exist any longer then the tester changes the status of the defect to 'Closed'.

Few More:

- **Rejected:** If the defect is not considered as a genuine defect by the developer then it is marked as 'Rejected' by the developer.
- **Duplicate:** If the developer finds the defect as same as any other defect or if the concept of the defect matches any other defect then the status of the defect is changed to 'Duplicate' by the developer.
- **Deferred:** If the developer feels that the defect is not of very important priority and it can get fixed in the next releases or so in such a case, he can change the status of the defect as 'Deferred'.
- **Not a Bug:** If the defect does not have an impact on the functionality of the application then the status of the defect gets changed to 'Not a Bug'.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

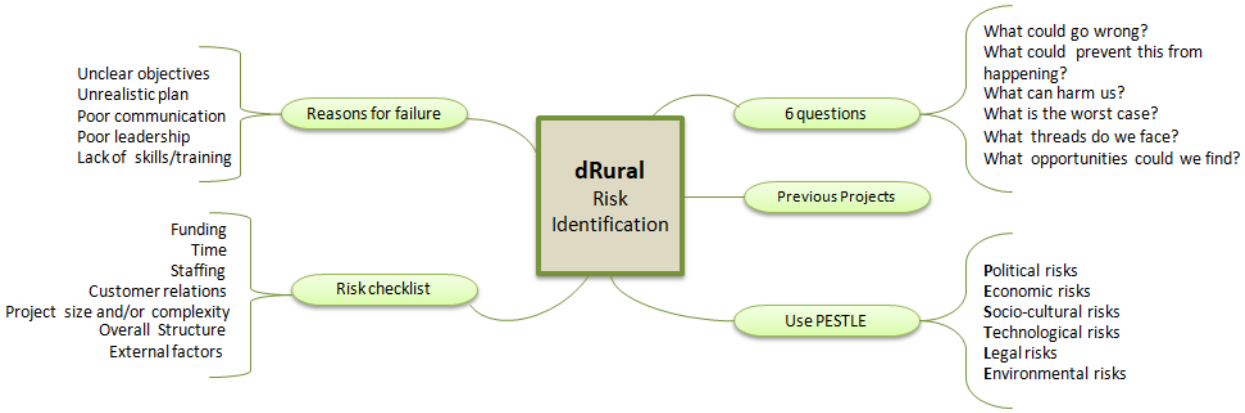
5 Risk Identification

Risk is an uncertain event or set of events that, should it occur, will have an effect on the achievement of objectives. A risk is measured by a combination of the probability of a perceived threat or opportunity occurring, and the magnitude of its impact on objectives.

Perhaps the simplest and fastest way to identify risks is to ask some simple, open questions:

- What could go wrong?
- What could prevent this from happening?
- What can harm us?
- What is the worst case scenario?
- What threats do we face?
- What opportunities could we find?

Figure 5: Techniques for identifying risk



Risk identification checklist:

- Funding: will it be enough? are some areas under-estimated
- Time: delays may happen, deadlines may change, deadline might be fixed
- Staffing: availability of skills, availability of key people, loss/reassignment of staff
- Customer relations: customer availability, knowledge of customer resources, customer maturity
- Project size and/or complexity: dRural is a large project and as such its monitoring and control is not easy.
- Overall structure: Is the project organization clear? Is it clear who has accountability? Are reporting lines clear?
- External factors: new regulations, market changes, changes in technology, changes in regulatory funding or processes.

The deployment process is inherently risky. Although in **dRural** the initial deployment is about new processes, after the first deployment will imply changing a *working*, *stable* version of the application to something we don't know will work. This can be intimidating, even for developers working and deploying code. And when working in a team with different stakeholders, as we do in dRural, the variability increases.

Like any business, **dRural deployment**, also possess entrepreneurial risks.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

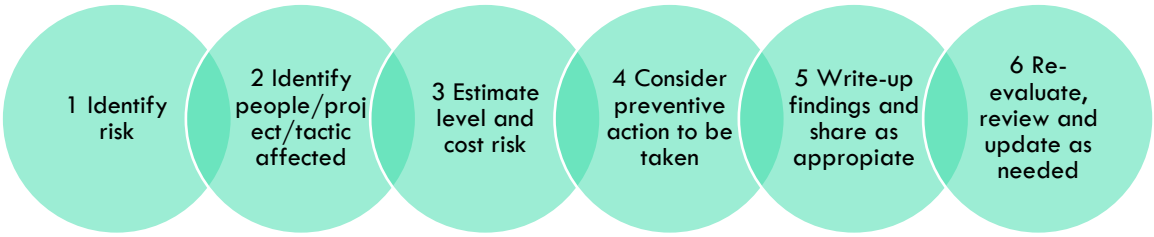
The objective of risk identification is to understand what is at risk within the **dRural** platform deployment context, and to generate a comprehensive inventory of risks based on the threats and events that might prevent, degrade, delay or enhance the achievement of the project objectives.

We have made a deliberate and systematic effort to identify and document **dRural**'s key risks (although there are many types of risks (specific of marketplaces, specific of tech companies, general risks, etc. there are not so many about deployment, being the main one to deploy in the wrong environment: production instead of development, but in any case they have a great impact

The risks identified cover the main deployment risks. The list includes the perspectives of all the partners, especially demonstrators and it will be updated at least once a year to consider new and emerging risks.

The following image synthesises the main steps of risk analysis

Figure 6. main steps of the risk analysis



5.1 Impact & Mitigation Actions

Although there are different types of risks (compliance, economic, financial, competitive, etc.) we focus on specific deployment risks. However, the below table can be applied indistinctively to any of them.

Figure 7. Risks Matrix

		Impact				
		Negligible 1	Minor 2	Moderate 3	Major 4	Catastrophic 5
Likehiood	5 Almost certain	Moderate 5	High 10	Extreme 15	Extreme 20	Extreme 25
	4 Likely	Moderate 4	High 8	High 12	Extreme 16	Extreme 20
	3 Possible	Low 3	Moderate 6	High 9	High 12	Extreme 15
	2 Unlikely	Low 2	Moderate 4	Moderate 6	High 8	High 10
	1 Rare	Low 1	Low 2	Low 3	Moderate 4	Moderate 5

Risk = Likelihood x Impact

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

5.1.1 dRural Deployment Specific Risks List

- Deploying to the Wrong Environment.

  It seems to be self-evident that before deploying to a production environment, the application has been tested and checked in the development environment and only after everything is OK is release to deploy. In dRural developer and deployer are different and deployer can only deploy after getting the version released by developer following strict testing procedures.

  Even so, sometimes the new version may contain an unstable GUI (elements popping from different places on the screen) or even an unusable one (no elements are present on the screen). The final test is done in real environments

- Deploying Prematurely

  It has been deployed to production correctly, but the features themselves are not yet ready. In this case, it would be needed to do a quick rollback. Setting a clear protocol for deploying to production mitigates this risk. The protocol should include the features' definition of done, manual QA, documentation, and every other check that the code should pass before it can be marked as safe to deploy to production. This checklist ensures that only complete features and working code will be deployed to production.

- Deploying the Wrong Code

  This is something that can happen. Sometimes a developer deploys the wrong branch of the repository. To mitigate this risk, it is a good practice to define which branches are going to be deployed to production and block the rest of the branches to prevent to go to production, preferably automatically

- Too Many User Scenarios

  Although we have progressed sequentially from development to production and made sure that our features work, manually checking all the possible user journeys is practically impossible in some scenarios. By "user journey," we mean the different permutations of GUI screens the user can encounter when using an application. For instance, an application can have different login screens based on the user type (such as corporate vs. private), different available features per the user's billing plan, different color schemes per the user's personal preferences, etc. Most people can't manually test such an overwhelming number of test cases. Moreover, even if it's possible, it's probably not cost-effective. In such scenarios it is convenient to use an automated tool to automatically check the different user journeys and make sure they all work as expected.

- Not Deploying at All

  This is possible. You can get a notification that everything is OK, but actually, nothing happens. The engineer didn't deploy any code. This can occur due to a bug in the deployment pipeline. To mitigate this risk, the engineer must ensure that they have a rock-solid production deployment pipeline. Performing manual QA (Quality Assurance [2]) on the production

[2] Quality Assurance methodology has a defined cycle called PDCA cycle or Deming cycle. The phases of this cycle are: Plan, Do, Check, Act

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

environment and ensuring that everything is there, can be a somewhat trivial solution, but it works.

Although "not deploying at all" might not sound like a deployment risk, it can be one of the most serious ones, since failing to deploy can be costly when deployments are time-critical and no one can reach the site.

- Crashing the Current Code Before the New One Is Up

  The current code can crash before the new code is active if we take the current environment down too soon. In this scenario, the current code's production environment can go down for a few minutes or more. If this happens before the new code is up, no one can access the site. To avoid this, we can utilize blue/green deployments or use an orchestrator to do the "switch."

- Leaving the Previous Version Online

  This happen when we don't take down the previous version after deploying the new code to production. In this case we have two production environments: one with the new code and one with the old code. Sometimes the old environment is given a different URL (such as old.site.com), and sometimes it has the same URL. The latter scenario guarantees that some customers will use the new version while others will use the old one. To avoid this situation, we need to manually take down the old version or via an automatic mechanism.

- Lack of familiarity with environments

  Having a deep understanding of the actual architecture and sizing requirements of each component (including external dependencies), is critical. Embracing different environments deployment to serve partners preferences may result in a failed run due to incorrect or inadequate architecture, especially when having to resolve findings in the given timeframe.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

5.1.2 dRural Deployment Specific Risks Mitigation Actions

Table 9: Deployment Specific Risks

	Risk	Probability (in %)	Impact (1-5)	Risk Evaluation (ProbxImpact)	Mitigation
1	Deploying to the Wrong Environment	5% - Rare	1 - Negligible	Low	Avoided as pilot site instances are fully separated at physical and management levels. Every pilot site "owner" can only deploy versions on its instance.
2	Deploying Prematurely	5% - Rare	2 - Minor	Low	Avoided as developer partner is the one releasing versions at the proper moment. Then, deployer partner collects the version and prepares the deployment specific Deployment Guide (DG) for pilot site to follow, detailing a clear step-by-step deployment protocol.
3	Deploying the Wrong Code	5%- Rare	1 - Negligible	Low	Avoided as deployer partner can only use versions released by developer partner.
4	Too Many User Scenarios	20%- Possible	3 - Moderate	High	Mitigated by checking the different user journeys and making sure they all work as expected.
5	Not Deploying at All	10%- Unlikely	4 - Major	High	Engineer(s) must ensure that they have a rock-solid production deployment pipeline
6	Crashing the Current Code Before the New One Is Up	5%- Rare	4 - Major	Moderate	Avoided by using only validated Deployment Modalities, using an orchestrator to manage the entire deployment procedure.
7	Leaving the Previous Version Online	5%- Rare	4 - Major	Moderate	Avoided by using only validated Deployment Modalities, using an orchestrator to manage the entire deployment procedure.
8	Not overcoming correctly the difficulty of adopting different forms of deployment	10%- Unlikely	3 - Moderate	Moderate	Prepare a deployment reference secure model, especially for on-premise environments

Probability: 0-5%: Rare; 5-10%: Unlikely; 10-20%: Possible 20-60%: Likely; >60%: Almost certain

Impact: 1:Negligible; 2: Minor; 3: Moderate; 4: Major 5: Catastrophic

Risk evolution: Low, Moderate, High, Extreme

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

6 Evaluation

6.1 Deployment Evaluation

Evaluations are critical to achieve deployment goals, are designed to determine the effectiveness and benefits of the solution deployed and are critical to an understanding of the value and effectiveness of the work done, allowing for the continual refinement of **dRural** strategy.

For deployment evaluation or assessment we will use specific KPIs as the way to measure performance against settled goals. KPIs help to know how close or far we are to achieving an objective and monitoring their progress allows us to correct the course of actions to achieve the strategic goals.

Ideally, **Metrics and KPI's**[3] present an overview of the deployment and change process — and where improvements can be made.

Whenever possible we will observe the following metrics to strive to improve both efficiency and user experience.

Figure 8. Deployment Metrics & KPIs

Relevant findings on Deployment Evaluation will be gathered and compiled in deliverables D4.3 to D4.6 (post-deployment reports) and shared with WP3 in order to introduce improvements in upcoming versions of the dRural platforms.

6.2 dRural Metrics and Key Performance Indicators

1. Deployment Frequency

Deployment frequency denotes **how often new features or capabilities are launched.** It can provide critical insights into the workflow and offers data-driven goals to improve upon.

Ideally, frequency metrics will either remain stable over time or see slight and steady increases.

[3] KPIs act as measurable benchmarks against defined goals. Metrics are different in that they track (or measure) the status of a specific business process. In short, KPIs track objectives/targets, and metrics track processes.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

Any sudden decrease in deployment frequency could indicate bottlenecks within the existing workflow.

More deployments are typically better, but only up to a point. If high frequency results in increased deployment time or a higher failure rate, it may be worth holding off on deployment increases until existing issues can be resolved.

2. Change Volume

This **KPI** determines the extent to which code is changed versus remaining static. Improvements in deployment frequency should not have a significant impact on change volume.

3. Deployment Time

It is about how much time is needed to roll out deployments (after they have been approved).

If deployments are quick to implement can occur quite often, but while short deployment time is essential, it shouldn't come at the cost of accuracy

Increased error rates may suggest that deployments occur too quickly.

4. Deployment Failure Rate

This metric determines how often deployments prompt outages or other issues, and this number should be as low as possible. The failed deployment rate is often referenced alongside the change volume. A low change volume alongside an increasing failed deployment rate may suggest dysfunction somewhere in the workflow.

5. Change Failure Rate

Allows to evaluate the deployment efficiency.

The change failure rate refers to the extent to which releases lead to unexpected outages or other unplanned failures. A low change failure rate suggests that deployments occur quickly and regularly. Conversely, a high change failure rate suggests poor application stability, which can lead to negative end-user outcomes.

CFR is calculated as a number of deployments where something went wrong (failure) divided with an overall number of deployments in a certain period.

6. Time to Detection

A low change failure rate doesn't always indicate that all is well with your application.

Mean time to detect (or MTTD) refers to the time it takes from when a problem first emerges to the moment when it is detected by the right people or systems.

High time to detection could prompt bottlenecks capable of interrupting the entire workflow.

The MTTD is measured by adding the incident detection time period and dividing it by the number of incidents. Example: time to detection: 600 minutes, incidents reported: 10. Mean time to detect would be 60 minutes. These KPIs can determine whether current response efforts are adequate

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

7. Mean Time to Recovery

It shows the efficiency to respond and solve problems that emerge along the way.

MTTR tells how much time is needed to solve issues and get back on track once failed deployments or changes are detected. This metric's value should decrease over time and should spike only in situations when facing problems that haven't been encountered before (adding new features, code complexity, changes in the operating environment or configuration....).

8. Lead Time for Change

Tells how much time is needed to implement a change. As the development cycle is a long process that will always require changes to occur, this KPI is one of the essential key performance indicators to monitor.

With all values for each issue, we take the *average*, the *median*, the *maximum* and the *minimum* lead-time.

There is no default threshold, but keep the value low. Automation can shorten the lead time.

9. Defect Escape Rate

Every software deployment runs the risk of sparking new defects and these might not be discovered until acceptance testing is completed.

Errors are a natural part of the development process and should be planned for accordingly. The defect escape rate reflects this reality by acknowledging that issues will arise and that they should be discovered as early as possible.

The defect escape rate tracks how often defects are uncovered in pre-production versus during the production process. This figure can provide a valuable gauge of the overarching quality of software releases.

10. Defect Volume

This metric relates to the escape rate highlighted above, but instead focuses on the actual volume of defects. While some defects are to be expected, sudden increases should spark concern. A high volume of defects for a particular application or process, may indicate issues with development or test data management.

11. Availability

Availability highlights the extent of downtime for a given application.

This can be measured as complete (read/write) or partial (read-only) availability. Less downtime is nearly always better. That being said, some lapses in availability may be required for scheduled maintenance. Track both planned downtime and unplanned outages closely, keeping in mind that 100 percent availability might not be realistic.

Achieving business continuity is a primary concern for modern organizations. Because every business is highly dependent on the Internet, every minute counts. That is why company computers and servers must stay operational at all times Downtime can cause significant financial impact and, in some cases, irrecoverable data loss.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

The solution to avoiding service disruption and unplanned downtime is employing a high availability architecture.

Availability is measured by how much time a specific system stays fully operational during a particular period, usually a year. It is expressed as a percentage. The uptime is usually expressed by using the grading with five 9's of availability. In a hosted solution, availability is defined in the Service Level Agreement (SLA) [4]

Uptime does not necessarily have to mean the same as availability. A system may be up and running, but not available to the users. The reasons for this may be network or load balancing issues.

12. Service Level Agreement Compliance

To increase transparency, most companies operate according to Service Level Agreements (SLA). A SLA is an agreement between an internal or external service provider and the end-user of that service. A SLA should clearly outline in simple language what the client will receive and what should be expected of the service provider.

SLA compliance KPIs provide the necessary accountability to ensure that SLAs or other expectations are met.

Although there is some overlap between SLAs and KPIs, they are different. Depending on the service, the types of metric to monitor by an SLA may include: Availability, Defect rates, Technical quality, Security, Business results... (this will depend as long as the vendor's contribution to those KPIs can be calculated).

Usually there are three different categories of service level agreements. They include:

- **Service-based**: The terms of service customers can expect are similar for all customers on a service-based service level agreement. In this case, everyone working with a service provider receives similar terms.
- **Customer-based**: A more customised service level agreement is customer based. This SLA outlines a relationship between a vendor and a customer in detail and is likely not a one-size-fits-all agreement.
- **Multi-level**: This category of SLA agreement splits into different levels to address a different set of customers who are using the same service.

It's important that SLAs include meaningful measurements so both the service provider and the customer can clearly assess performance. It is here where some overlap occurs between SLAs and KPIs.

[4] A **service level agreement** is a legal contract between a provider and the end user. Its primary purpose is to make sure that both parties involved agree on the services to be provided and the standards to which adhere.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

6.3 User Evaluation

WP4 does not cover business aspects or impact analysis, being WP6 the one dealing with those aspects and WP4 evaluation will only evaluate if the deployed technology meets/aligns to the user requirements defined in WP3.

From a deployment perspective, **dRural** will evaluate the tool's technical capabilities, as well as the potential for errors and difficulties involved in using it. Usability is related to development affecting inevitably the deployment. And Usability of the solution can be only known after the feedback received from end-users. Users' satisfaction, understood from a technical perspective, will let us know if the deployment goals have been reached or not.

Approach to evaluation: A technical dimension looking at the platform and service performance (Usability)

Besides reporting bugs, for evaluation, we will combine quantitative and qualitative techniques, as a "mixed" evaluation method, with the aim to obtain a richer and more comprehensive understanding of project's results.

Qualitative: is descriptive, and regards phenomenon which can be observed but not measured

Quantitative measures the depth and breadth of an implementation (e.g., the number of people who participated, the number of people who completed the program), but does not provide an understanding of the context and may not be robust enough to explain complex issues or interactions.

Table 10: dRural Qualitative and Quantitative evaluation methods (from a technical perspective)

Methods	Tools	Description
Quantitative: numerical data compiled arithmetically and analyzed by statistical processes	Surveys/ Questionnaires	Series of questions that generate information or opinions in a numerical form to be analyzed
	Existing data and statistics	Existing figures and statistics, articles and books already published, in relation to demographics and Rural Communities in different Member States
	System log files	Monitoring through dRural platform (Dashboard) the technical performance of the overall system and pilot performance.
Qualitative: Information gathering from interviews, observations and other sources that require analysis through interpretation and inference	Interviews [Number completed]	Structured/unstructured questioning Semi-structured questioning, with pilots leaders and participating organisations and other policy experts
	Surveys/ Questionnaires [Number completed]	*Series of questions that generate information or opinions in a contextual form* Professionals Survey; Users enrolled in the pilot Survey
	Observations [Number / Hours]	Focused observations (on meetings and users interactions / attitudes) To understand things at a deeper level

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

	Internal Audit	A summary on subjective observations of the pilot will be made by pilot responsible and provided to Coordination

Quantitative data collected will be consistent and reliable enough and easy to analyse, while the findings can be generalised, following a user sample selection representative of the pilot users' profile.

dRural will combine the quantitative with qualitative methods, especially from information gathering from interviews, observations and other sources that require analysis through interpretation and inference. Local pilots will have internally to interact through discussion, interviews and observations, periodically with the local focus groups formed at each of the pilots' sites and spend time with them, eliciting their thoughts and opinions either directly or indirectly.

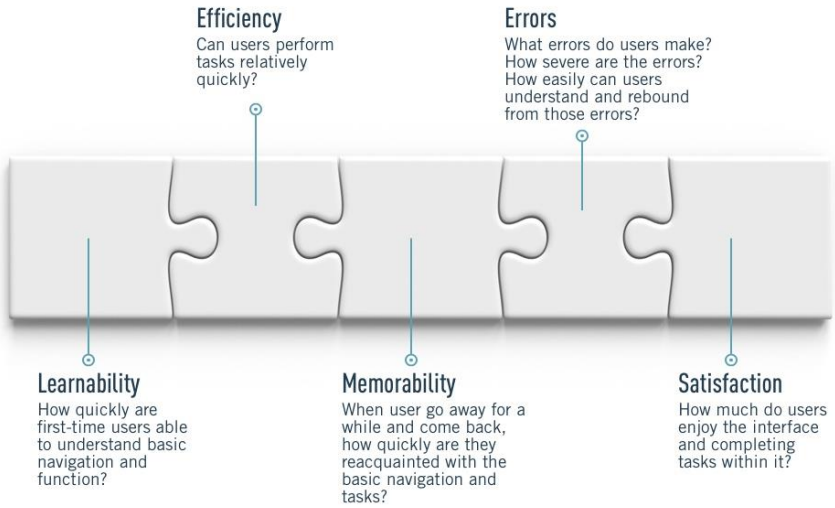
With the qualitative methods we can interpret the meanings that people assign to their requests and demands, on the basis of their everyday life needs, providing more detailed information to explain complex issues.

6.4 Usability

Usability focuses on how easily people can finish a particular task. Under ISO 9241-11 standard, it is defined as "the effectiveness, efficiency, and satisfaction with which specified users achieve specified goals in particular environments".

Usability is based on five core qualities: learnability, efficiency, memorability, errors, and satisfaction.

Figure 9 Usability qualities



- **Learnability.** How quickly are first-time users able to understand basic navigation and functions?
- **Efficiency.** Can users perform tasks relatively quickly?
- **Memorability.** When users go away for a while and come back, how quickly can they reacquaint themselves with the basic navigation and functions?
- **Errors.** What errors do users make? How severe are the errors? How easily can users understand and rebound from those errors?

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

- **Satisfaction.** How much do users enjoy the interface and completing tasks within it? (Both as they are using the product and how they report satisfaction afterwards.)

Measuring a Product's Usability with SUS

The System Usability Scale, also known as SUS, is a simple 10-item questionnaire designed to measure people's perceptions of usability. Although it's ten questions long, the SUS survey uses simple scale questions (5-point scale) that are quick and easy for respondents to follow.

Each of these questions asks users to rate their agreement from "Strongly Disagree" to "Strongly Agree":

Table 11: Usability template

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
	1	2	3	4	5
I think that I would like to use dRural frequently					
I found dRural unnecessarily complex					
I thought dRural was easy to use					
I think that I would need the support of a technical person to be able to use dRural.					
I found the various functions in dRural were well integrated					
I thought there was too much inconsistency in dRural					
I would imagine that most people would learn to use dRural very quickly					
I found dRural very cumbersome to use					
I felt very confident using dRural					
I needed to learn a lot of things before I could get going with dRural					

This set of questions focus solely about the users experience using the product itself.

Nowadays it is commonly used for all sorts of products, including hardware, websites, cell phones, and other physical products.

The System Usability Scale is quick and inexpensive to implement. Its results have consistently proven valid and useful

How to Calculate the Usability Score using SUS

The statements that are odd (1, 3, 5, 7 and 9), you have to subtract 1. (For example statement 5 has been evaluated as 3, **3 –1 = 2**, you'll compute 2)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

For the even statements (2, 4, 6, 8, 10), you have to subtract from 5. (Let's suppose that the value for the 4th statement was 3. Then we have **5–3=2**, so you'll compute 2)

At the end you'll have to sum all these numbers and multiply by 2.5.

The average System Usability Scale score is 68. If the score is under 68, then there are probably serious problems with usability that need to be addressed. If the score is above 68, then it is ok although could be better.

Here's an overview of how scores should measure:

- 80.3 or higher is an A.
- 68 or thereabouts gets a C. It's OK but could improve
- 51 or under gets a big fat F. It is not good and Usability should be fix asap

As for Satisfaction or how happy users are with the service (overall experience either falls short, meets, or exceeds their expectations), although not considered a technical issue and therefore outside the scope of deployment, should, be part of the outcomes being dealt with by WPs 6 and 8, since, in our understanding, technical outcomes (reflected mainly in Usability) are key elements adding for satisfaction.

For measuring satisfaction, usually three main metrics are used in combination: Customer Satisfaction Score (CSAT); Net Promoter Score (NPS) and Customer Effort Score (CES).

Of them CES (Customer Effort Score) metric or effort required to achieve an intended outcome, is purely transactional and because its appropriateness for a technical outcome (Usability), should be controlled.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

7 Ethics, Privacy and Data Protection

GDPR compliance has become crucial for business running on marketplaces.

dRural follows a transparent policy privacy and any possible personal data collected is with the only purpose of carrying out the project activities.

The data collected in **dRural** will be anonymised and only aggregated data will circulate (when needed), outside the promoter.

dRural needs to ensure the business ecosystem is secure. At the very least, marketplace operators are responsible for jointly processing personal data with third-party services on the platform.

The activities to be undertaken at the level of pilots, concerning ethics, privacy, and societal acceptability, do not raise any issue needing to be declared, however, even if pilots data are design protected and used with the sole purpose of carrying out the project activities, inform consent will be required from the demonstrators partners, to process any personal data they may use and proving informed consent has been given freely, with no pre-ticked boxes allowed.

Demonstrators have to tell users which types of data are collected and the reasons why, explaining what it will be used for.

As for new users they can give their consent and see the place policies before proceeding to buy something from the platform.

No personal data is retained by technical partners developing and/or implementing the platform, since the platform is hosted in the cloud. The cloud provider(s) to work with is/are based in the European Economic Area and is/are GDPR compliant, and equipped with the highest security measures. And if on-premises, data responsibility will be of the partner carrying out the operations.

Data when travelling between users and providers is encrypted using the standard TLS (Transport Layer Security). Technical partners use suitable technical and organisational safety procedures to protect data against inadvertent or wilful manipulation, partial or complete loss, destruction or against the unauthorised access by third parties.

Initially there are not subcontractors in **dRural**, meaning we have not to face (at least by now), this issue.

And, finally, there is the payment service provider(s). **dRural** only partners with business ensuring conformity to the laws.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.

8 Conclusion

The plan outlined above depends on the proper implementation and willingness to make adjustments as and when necessary. Each demonstrator is responsible for carrying out the prescribed functional tactics to put the generalized strategy into specific actions in order to achieve the project's objectives. Deployer partner will do any required efforts to achieve the proper functioning of the solution and will focus on helping and supporting demonstrators as needed.

Deployment will be offer to the regions with different alternatives: from the Cloud and On-Premise.

Operating from the cloud, has raise concerns in some of the demonstrators, already widespread across Europe that because of the GDPR, data cannot be hosted by a subsidiary of a U.S. company.

For partners peace of mind, there is to say there are favourable judgments ruling such issue when for the purposes of hosting the data, are being used services from companies located in Europe and the level of protection of data is appropriate, this is, that sufficient safeguards, both legal and technical, are in place.

Moreover, the EC-approved Standard Contractual Clauses (SCCs), adopted in June 2021 give cloud providers and their customers the ability to comply with the General Data Protection Regulation (GDPR) when they transfer personal data subject to GDPR to countries outside the European Economic Area (EEA) that haven't received an EC adequacy decision (third countries). Be aware there are many companies transferring data to third countries

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017304.