SOFIE - Secure Open Federation for Internet Everywhere

779984

DELIVERABLE D3.3

Business Platforms Pilot Release

Project title	SOFIE – Secure Open Federation for Internet Everywhere
Contract Number	H2020-IOT-2017-3 – 779984
Duration	1.1.2018 – 31.12.2020
Date of preparation	30.9.2019
Author(s)	Santeri Paavolainen (LMF), Mikael Jaatinen (LMF), Panu Koskela (LMF), Antonio Antonino (LMF)
Responsible person	Mikael Jaatinen (LMF), mikael.jaatinen@ericsson.com
Target Dissemination Level	Public
Status of the Document	Completed
Version	1.00
Project web-site	https://www.sofie-iot.eu/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 779984.

Table of Contents

1 Introduction

SOFIE is a three-year EU Horizon 2020 research and innovation project with the goal to enable diversified applications from various application areas to utilise heterogeneous IoT platforms and autonomous things across technological, organisational and administrative borders in an open and secure manner, making reuse of existing infrastructure and data easy.

IoT business platforms are created in SOFIE, based on the IoT federation framework defined in WP2. For this, an IoT framework repository, consisting of various components, adapters for well-known IoT platforms and security mechanisms is developed. These components can be used to create business platforms, including those for the four SOFIE real-world pilot use cases.

During 2018-2020, SOFIE will deliver three business platform main releases as defined by deliverable “D3.1 - Integration Plan” [Jaa2019]. Within every main release, new functionality is added through a Continuous Integration (CI) and Continuous Deployment (CD) process. The CI and CD processes are provided via the WP3 CI/CD environment.

The first Lab Prototype Release was made available in November 2018 as described by deliverable “D3.2 - Business Platform, Lab Prototype Release” [Jaa2018] with continuous functional growth through monthly minor releases.

The present deliverable (D3.3 - Business Platform, Pilot Release) builds further on top of “D3.1 - Integration Plan” [Jaa2019] and “D3.2 - Business Platform, Lab Prototype Release” [Jaa2018] by providing an environment (tools) and methodology for integrating SOFIE Framework components available in public SOFIE repositories and specified by “D2.5 - SOFIE Framework, 2nd version” [Kor2019] as well as pilot-specific components in private repositories. For details on the Business Platform, Pilot Release scope and contents, please see chapters 2 and 3.

A testbed and Emulation environment, as described by deliverable “D4.2 - Testbed and Emulation Environment Design and Setup” [Lag2019] is available to support integration testing. The SOFIE Pilot deployments leverage on the Business Platform, Pilot Release in accordance with “D5.1 - Baseline System and Measurements” [Oik2018] and “D5.2 - Initial Platform Validation” [Oik2019]. Please see chapter 4 for more details on these related deployments.

Figure 1 depicts the relation between the Business Platform, Pilot Release (D3.3) and relevant related SOFIE project deliverables.

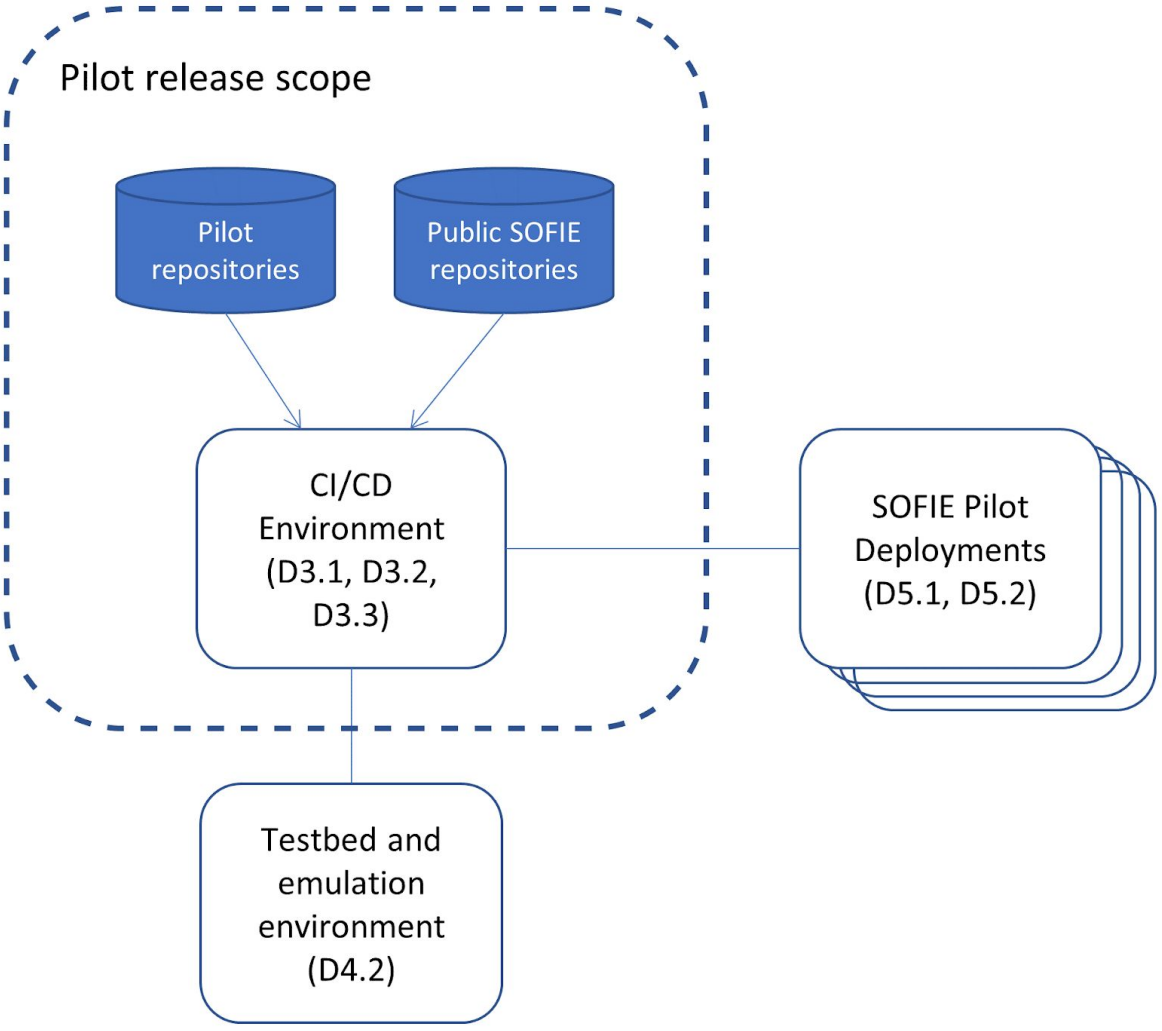

Figure 1. Relevant deliverables related to Business Platform, Pilot Release

2 Release Information

2.1 Release identifier

Release name: **SOFIE Business Platform Pilot Release (version 1)**

Release date: 30th of September 2019

2.2 Release scope

The Business Platform Pilot release is based on the SOFIE Deliverables up to end of September 2019 and is integrated in accordance with deliverable "D3.1 - Integration Plan" [Jaa2019].

The release scope consists of:

- WP2 Framework components that are available in public software repositories
- WP5 Pilot components for which the software repositories have been integrated to the WP3 CI/CD environment
- The WP3 CI/CD process and methodology
- The WP3 CI/CD environment for business platform integration

The Business Platform Pilot Release will be continuously enriched with new features as new software components are onboarded to the WP3 CI, existing software components are enriched and the WP3 CI/CD environment is further developed.

A WP3 CI/CD onboarding maturity level has been defined for SOFIE software components that are part of the Business Platform Pilot Release. The different levels are specified in Appendix I. The onboarding status as of the release date is provided in Section 3.1.

2.3 Purpose and target users

The purpose of the Business Platform Pilot release is to:

- Deliver the available SOFIE software components for evaluation in WP4 in accordance with "D4.1 - Validation and Evaluation Plan" [Sir2018]
- Integrate available SOFIE software components for WP5 pilot field trial usage in accordance with "D5.1 – Baseline system and measurements" and "D5.2 - Initial Platform Validation" [Oik2018,Oik2019]
- Provide quality control mechanisms for the WP2 Framework components that are available in public software repositories for anyone to use
- Deliver a complete CI environment in the AWS public cloud for SOFIE users
- Deliver the CI/CD processes and methodology for SOFIE users

3 Release Contents

3.1 Components

One of the reasons of providing CI-as-a-Service to the developers of different pilot software components and framework components is to increase the overall quality of the software components produced. By owning and managing the CI environment, LMF Ericsson is able to take advantage of its expertise in the field and offer a set of additional services that might not be available to the pilots in their testing environments, thus increasing the overall quality of the components code, e.g. by reducing the number of bugs that might affect the functionalities of the component, once made publicly available to its users. In the same fashion, by hosting the entire CD runtime, LMF offers a Platform-as-a-Service environment by deploying all the frameworks and pilots components on the infrastructure it owns and manages.

Several components have been planned to be integrated into the CI/CD pipeline. Some of the components are developed by SOFIE pilots. Others are developed by WP2 to either be integrated into the different pilots, upon proper customisation depending on the needs of each pilot, or to be deployed as standalone components.

All the components that have been released by the project, along with their current status, are described in Table 1 below. Each component is detailed with regard to the following fields:

- **Origin**: the SOFIE partner leading the development of the component.
- **Component Users**: the pilots that have planned to use the component by integrating it into their codebase.
- **Integration level**: the level of maturity of the component with regard to the CI/CD pipeline. Appendix I describes the meaning of each maturity level and the actions that must be taken by the component developers to take the component to each maturity level.
- **Public**: whether the source code of the component is publicly accessible or not.
- **Description**: the URL of the README file of the component source code repository for publicly available components.

The table below reflects the components as of the release date of this deliverable.

Table 1. List of SOFIE project software components, including their originator, which pilot(s) use the component, the CI/CD maturity level and a link to the component README in the SOFIE project source code repository. The users of the components are Food Supply Chain Pilot (FSCP), Decentralized Energy Data Exchange Pilot (DEDEP), Decentralized Energy Flexibility Marketplace (DEFM) and Mixed Reality Mobile Gaming Pilot (MRMG).

Component	Origin	Component Users	Integration level	Public	Description
Discovery and Provisioning	ROVIO	MRMG	0	yes	README
DSO dashboard	ENG	DEFM	3	no	-
DSO backend	ENG	DEFM	3	no	-
IAA	AUEB	DEFM, FSCP	0	yes	README
Interledger Asset Transfer	AALTO	FSCP, MRMG	0	yes	README
Interledger demo	AALTO	-	3	no	-
Offer Marketplace	AALTO	DEFM	0	yes	README
Privacy and Data Sovereignty	AUEB	FSCP	0	yes	README
Semantic Representation	LMF	MRMG	0	yes	README
SOFIE Adapter Application	GT	DEDEP	3	no	-

3.2 CI/CD pipeline

On-boarding a component onto the CI/CD pipeline requires the cooperation between LMF and the developer of the component. Clear instructions have been provided by LMF about how to prepare a component to be on-boarded and how to properly write documentation for it in the integration documentation template file.

At any given time, a component can be in one of the seven maturity levels shown in Appendix I. Levels **1-3** are relative to the CI maturity, while levels **4-7** indicate the maturity of the component with regard to the CD environment. Any component must meet the full CI maturity level (level 3) before proceeding to CD integration (level 4 onwards).

Specifically for the CI maturity levels, component developers are responsible for:

1. Giving the CI agent read-only access to the component repository either via SSH-based (preferred) or HTTPS-based authentication.

2. Configuring webhooks to trigger new builds in the CI environment whenever new commits are pushed on a branch specified in the integration documentation. *The completion of steps 1 and 2 promotes the component to maturity level 1.*
3. Defining the component build process in a way that produces Docker images as final artifacts that are parametrized with regard to potential hard-coded IP addresses and port numbers. The tagging details of the Docker images must be specified in the integration documentation.
4. For those artifacts, writing unit tests that are collected in JUnit format in a path that must be specified in the integration documentation. *Fulfilment of steps 3 and 4 promote the component to maturity level 2.* Nevertheless, in case of failure of either the build process or the unit tests, the component cannot be taken to step 3, since the artifacts will not be pushed onto the remote artifact repository.

Taking a component from level 2 to level 3 is entirely the responsibility of LMF, which will create the required ECRs, or Elastic Container Registries, on AWS. The registries have the proper permissions so that only the LMF staff is able to access them in read-write mode and the component developers in read-only mode. Once the required ECRs are created, the CI build process pushes the artifacts there so that the CD pipeline can be triggered and can pull the new artifacts.

Even though maturity levels have been defined also for the CD pipeline, the process to take components from level 3 to level 4 and further has not yet been defined. They will be defined step-by-step by constantly exchanging feedback with the different developers to achieve the best trade-offs in terms of security and reliability of the CD environment.

3.3 CI and CD Infrastructure

The core capabilities the environment provides are:

1. Perform **continuous integration** tasks when triggered by changes in watched repositories of SOFIE WP2 and WP5 material. Supports multiple repositories and build pipelines.
2. Perform **continuous delivery** tasks, including automated integration tests, and deployment to staging and production environments. Supports multiple pipelines, e.g. for each pilot separately.
3. **Support per-pilot integration** needs by being able to provide servers for pilots to configure any necessary gateways etc. used during integration testing, staging and production deployments.
4. Enough flexibility in CI/CD and environment to make it reasonably feasible to automate various evaluation tasks, and/or manual validation and evaluation tasks.

The underlying infrastructure where the integration and evaluation environment is being deployed as part of WP3 is the **Amazon Web Services (AWS)** cloud environment. An EU region is used for the deployment (the UK is not used, due to the possibility of Brexit).

Figure 2 shows the interactions between LMF, the pilot components, and the CI/CD pipeline.

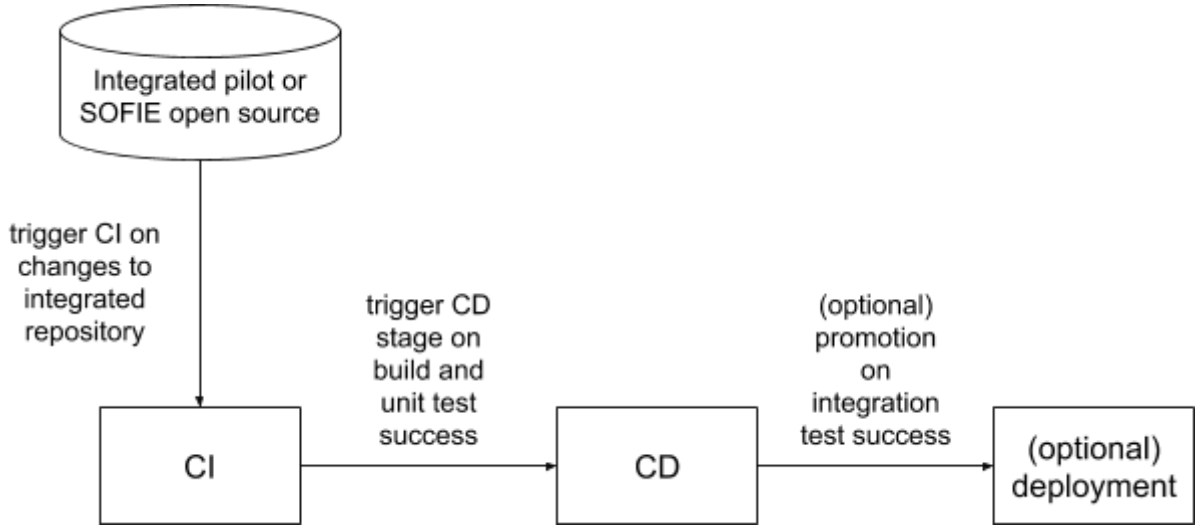

Figure 2. Overview of how WP2 and WP5 components are integrated and deployed using the CI/CD pipeline built by LMF. Changes to the integrated repository trigger a CI task that builds the repository and performs unit tests. On successful completion of the CI task, a CD task is triggered, which will perform integration tests. If integration tests are successful, a promotion to a persistent deployment may be performed if required.

3.4 CI architecture

Figure 3 shows an overview of the infrastructure environment components and their major relationships. This environment is not redundant and not designed for high availability, although it has been highly reliable so far. The underlying assumption is that while CI and CD processes are important, a failure due to the loss of a virtual machine, database etc. is an unlikely event, and, in the worst case, manual recovery through rebuilding of failed components will last only a week at most. The “infrastructure as code” approach for the environment setup allows easy and repeatable deployments of the complex multi-node integration environment. The Terraform tool is suitable for this purpose as it supports incremental deployments and parameterization of the deployments, including integration of externally managed resources into the deployment templates. In this model, the integration environment itself is described in the Terraform template language, and stored in a version-controlled source code repository. The templates contain specific environment information and may contain sensitive information, so these are not made public by default.

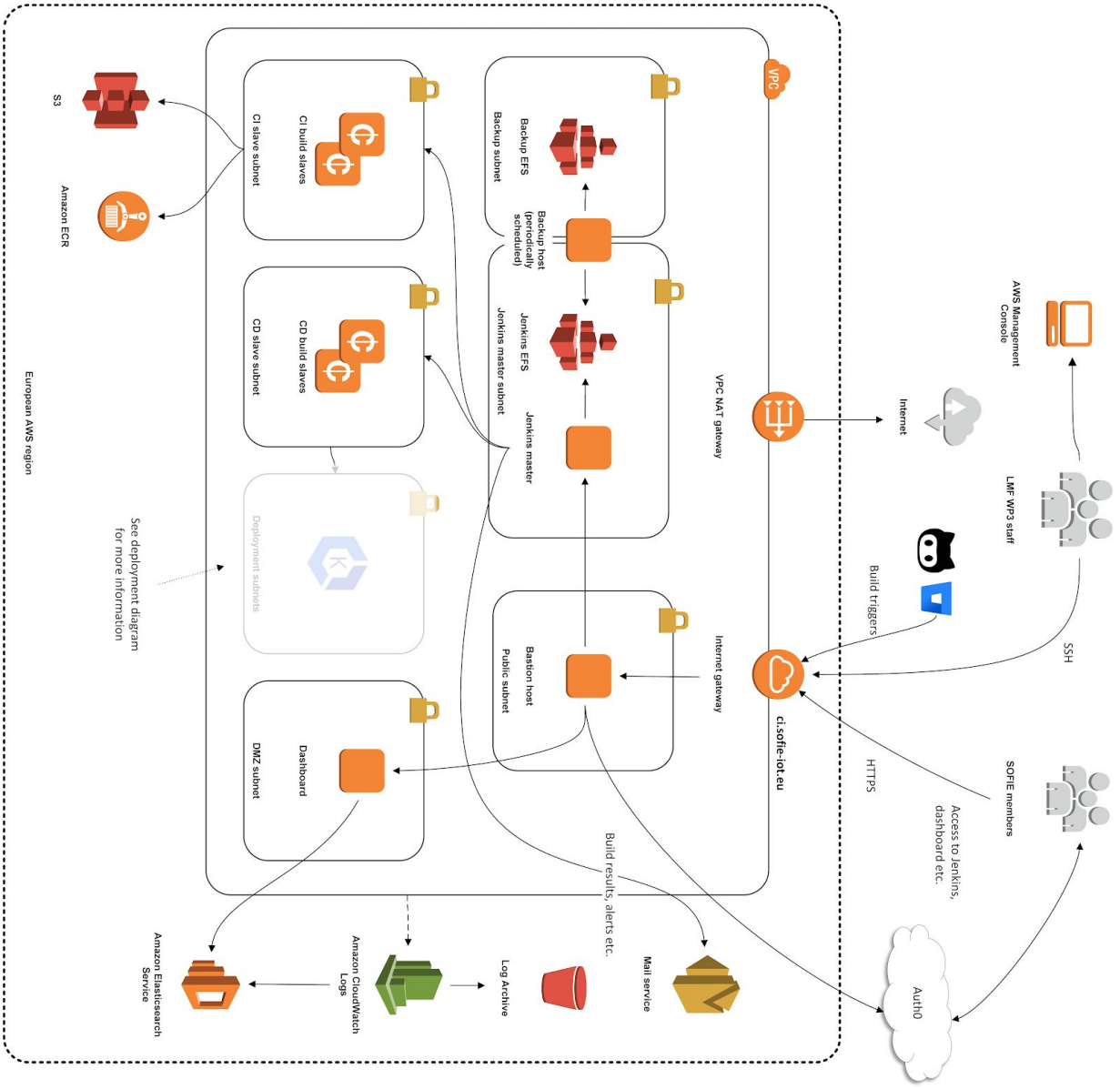

Figure 3. Overview of the CI architecture. The environment is run in an isolated network accessible only via the bastion host for authenticated Jenkins users and administrative staff. The deployment includes automated scheduled backups, centralized logging and monitoring, and dynamically scaled build fleet for CI and CD tasks.

3.5 CD architecture

Figure 4 shows the major components of the deployment environment. The deployment environment is separated from other infrastructure services (different subnets, different VPC, etc). The deployment process is driven by the Jenkins CD slave fleet and uses general Kubernetes deployment functions provided by AWS EKS. In the CI/CD environment access to the EKS subnets is limited to the CD slaves only.

Deployed services are run in Docker containers orchestrated by AWS EKS (Kubernetes). EKS consists of a separate master cluster (EKS control plane) and a cluster of worker nodes. The master cluster orchestrates the cluster and provides the Kubernetes API endpoint while the worker nodes run the containers. The EKS control plane is managed by AWS to provide automatic healing creating a fully production ready platform. The worker nodes are an EC2 auto scaling group, which enables automated scaling for the cluster.

In addition to Docker containers, deployments will be potentially supplanted by other temporary or persistent services (database, cache, pilot backend gateways, etc.).

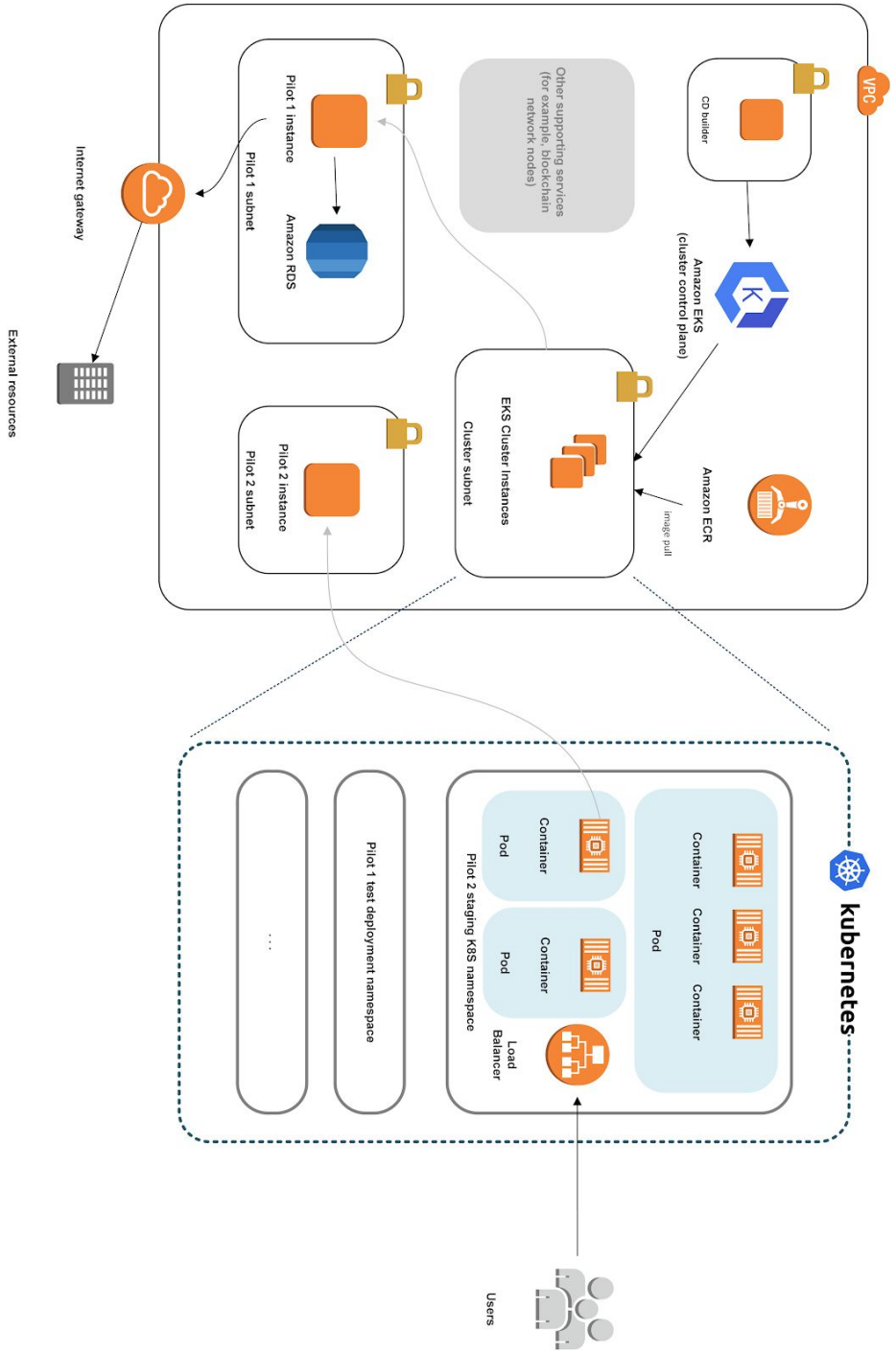

Figure 4. CD environment for testing, staging and production deployments. The testing and persistent deployments are done on a Kubernetes cluster on separate namespaces for logical and network isolation. The CD environment also contains any persistent resources required for either integration testing or persistent deployments including any custom gateways to external resources or persistent databases, for example. The deployment and integration testing process is controlled from the Jenkins master indirectly via the CD builder node. A persistent deployment may also include either restricted or publicly accessible resources.

4 Related Deployments

4.1 Testbed and emulation environment

A SOFIE testbed and emulation environment is available as described in deliverable “D4.2 - Testbed and Emulation Environment Design and Setup” [Lag2019].

The testbed spans multiple project partners (AALTO, AUEB, LMF Ericsson) and allows testing various distributed ledger technologies and their interaction with IoT devices on a wider scale. Testbed elements include private Ethereum, Hyperledger Fabric, Hyperledger Indy and access to Guardtime KSI blockchain.

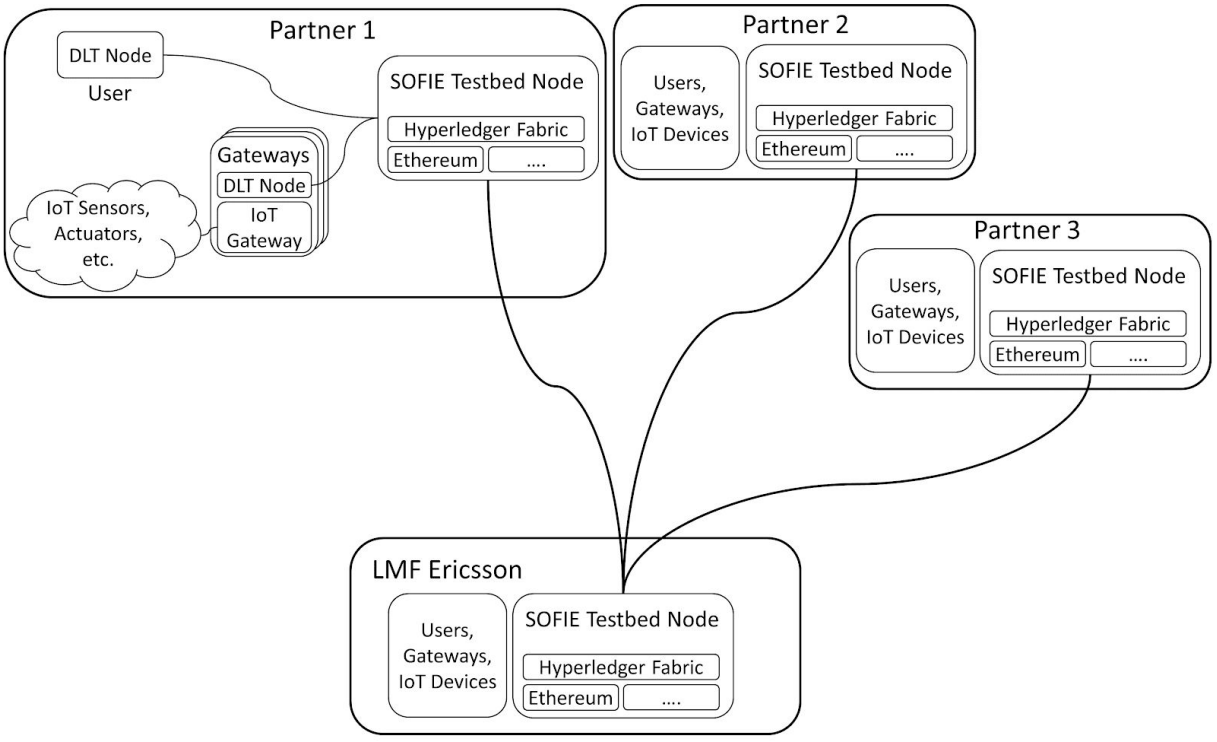

Figure 5. Overview of the SOFIE testbed setup

The SOFIE emulation environment emulates certain aspects of SOFIE pilots and related, more general, use-cases, thus allowing realistic testing of various solutions without deploying them yet in pilot environments.

The testbed and emulation environment is used as an additional means to validate the correct behavior of integrated business platforms, with a close feedback loop between WP3 and WP4. The local testbed at LMF Ericsson is hosted in the same Amazon Web Services account as the CI/CD environment, making it easy to use testbed components as part of CI and CD activities.

4.2 Pilot deployments

Before field deployment, a proof of concept prototype has been implemented and demonstrated in the lab environment for every SOFIE pilot, as described by deliverable “D5.2 - Initial Platform Validation” [Oik2019]. WP3 has supported the pilot proof of concept development with the CI and CD capabilities described in Section 3, and will similarly support the pilot field deployments as they are further developed.

5 Changes since previous release

The main changes since previous release:

- Introduction of new WP2 framework components
- Full CI capabilities available in the WP3 CI and CD environment
- CI and CD pipelines defined for SOFIE
- Onboarding and CI processes refined to facilitate faster onboarding through maturity levels 1 to 3

6 References

[Jaa2018] M. Jaatinen “SOFIE Deliverable D3.2 - Business Platform, Lab Prototype Release”, November 2018. Available at: https://media.voog.com/0000/0042/0957/files/SOFIE_D3.2-Business_Platform_Lab_Prototype_Release.pdf

[Jaa2019] M. Jaatinen, “SOFIE Deliverable D3.1 - Integration Plan”, April 2019. Available at: https://media.voog.com/0000/0042/0957/files/SOFIE_D3.1-Integration_Plan.pdf

[Lag2019] D. Lagutin et al., “SOFIE Deliverable D4.2 - Testbed and Emulation Environment Design and Setup”, February 2019. Available at: https://media.voog.com/0000/0042/0957/files/SOFIE_D4.2-Testbed_and_Emulation_Environment_Design_and_Setup-v1.00.pdf

[Oik2018] I. Oikonomidis et al., “SOFIE Deliverable D5.1 - Baseline System and Measurements”, June 2018. Available at: http://media.voog.com/0000/0042/0957/files/SOFIE_D5.1-Baseline_System_and_Measurements.pdf

[Oik2019] I. Oikonomidis et al., “SOFIE Deliverable D5.2 - Initial Platform Validation”, July 2019. Available at: https://media.voog.com/0000/0042/0957/files/SOFIE_D5.2-Initial_Platform_Validation.pdf

[Sir2018] V.A. Siris et al., “SOFIE Deliverable D4.1 - Validation and Evaluation Plan”, October 2018. Available at: https://media.voog.com/0000/0042/0957/files/SOFIE_D4.1-Validation_and_Evaluation_Plan-v1.00.pdf

[Kor2019] Y. Kortesniemi et a. “SOFIE Deliverable D2.5, Federation Framework, 2nd Version”, August 2019. Available at: https://media.voog.com/0000/0042/0957/files/SOFIE_D2.5-Federation_Framework%2C_2nd_version.pdf

7 Appendix I: Onboarding Maturity Levels

#	Level criteria	Requirements for component developers
Level 0	No integration to SOFIE CI/CD	-
Level 1	CI is triggered on new commits, pull repository (fetch source code)	1. Give *read-only* access to the repository to Jenkins (possibly SSH-based) 2. Configure a webhook for push events to the component-specific URL
Level 2	CI Build and unit tests pass, unit test results are collected correctly	1. Build produces properly parametrized and properly documented Docker images 2. At least one unit test is implemented and must complete with no errors. Test results are collected in JUnit format.
Level 3	CI Build artifacts are pushed to artifact storage (ECR, artifactory, others) and can be used by developers (e.g. pulled locally)	**Entirely WP3 responsibility**
Level 4	CD test environment, that uses the artifacts, can be deployed, and does not enter a crash-restart loop	TO BE DEFINED
Level 5	CD Some integration tests exist, and they pass on the CD test deployment	TO BE DEFINED
Level 6	CD Staging deployment can be deployed after successful integration test build (e.g. semi-persistent deployment)	TO BE DEFINED
Level 7	CD Promotion process and deployment for "production" deployment	-