OpenRiskNet

RISK ASSESSMENT E-INFRASTRUCTURE

Deliverable Report D2.6

Reference OpenRiskNet system available online

This project is funded by

the European Union

OpenRiskNet: Open e-Infrastructure to Support Data Sharing, Knowledge Integration and *in silico* Analysis and Modelling in Risk Assessment

Project Number 731075

www.openrisknet.org

Project identification

Grant Agreement	731075
Project Name	OpenRiskNet: Open e-Infrastructure to Support Data Sharing, Knowledge Integration and *in silico* Analysis and Modelling in Risk Assessment
Project Acronym	OpenRiskNet
Project Coordinator	Edelweiss Connect GmbH
Star date	1 December 2016
End date	30 November 2019
Duration	36 Months
Project Partners	P1 Edelweiss Connect GmbH Switzerland (EwC) P2 Johannes Gutenberg-Universität Mainz, Germany (JGU) P3 Fundacio Centre De Regulacio Genomica, Spain (CRG) P4 Universiteit Maastricht, Netherlands (UM) P5 The University Of Birmingham, United Kingdom (UoB) P6 National Technical University Of Athens, Greece (NTUA) P7 Fraunhofer Gesellschaft Zur Foerderung Der Angewandten Forschung E.V., Germany (Fraunhofer) P8 Uppsala Universitet, Sweden (UU) P9 Medizinische Universität Innsbruck, Austria (MUI) P10 Informatics Matters Limited, United Kingdom (IM) P11 Institut National De L’environnement Et Des Risques, France (INERIS) P12 Vrije Universiteit Amsterdam, Netherlands (VU)

Deliverable Report identification

Document ID and title	Deliverable 2.6 Reference OpenRiskNet system available online
Deliverable Type	Demonstrator
Dissemination Level	Public (PU)
Work Package	WP2
Task(s)	Task 2.6
Deliverable lead partner	UU
Author(s)	Ola Spjuth (UU), Tim Dudgeon (IM), Alan Christie (IM), Evan Floden (CRG), Atif Raza (JGU), Thomas Exner (EwC)
Status	Final
Version	V1.0
Document history	2019-09-09 First draft 2019-11-15 Consolidated draft 2019-11-28 Final version

Table of Contents

SUMMARY

In this deliverable we present the availability of the OpenRiskNet reference system. The system is online with operational functionality and demonstrated operations in the OpenRiskNet case studies as well as within the Associated Partner Program. This deliverable is in the form of a Demonstrator, and apart from describing the reference site and examples of applications, it also summarises the recent developments in WP2 that has not been reported earlier.

TASKS

The following is an overview and final status report of the tasks in WP2:

Task	Activities/Status/Deviations
2.1: Creation of development environment	A complete development environment was set up by M6 and has been reported in D2.1 [1].
2.2: API specification and semantic interoperability	This work was described and reported in D2.2 [2] and D2.4 [3]. **The recent updates regarding this task are reported in this document (D2.6).**
2.3: Establish security environment	This work was described and reported in D2.3 [4]. **The recent updates regarding this task are reported in this document (D2.6).**
2.4: Services discovery	This work was described and reported in D2.3 [4]. **The recent updates regarding this task are reported in this document (D2.6).**
2.5: Deployment of virtual infrastructures and container orchestration frameworks	This work was described and reported in D2.3 [4]. **The recent updates regarding this task are reported in this document (D2.6).**
2.6: Establishment and maintenance of OpenRiskNet reference instance	This work was described and reported in D2.3 [4]. **The recent updates regarding this task are reported in this document (D2.6).**
2.7: Interconnecting virtual environment with external infrastructures	This work was described and reported in D2.5 [5]. **The recent updates regarding this task are reported in this document (D2.6).**
2.8: Federation between virtual environments	This work was described and reported in D2.5 [5].

OPENRISKNET E-INFRASTRUCTURE, VIRTUAL ENVIRONMENTS AND SERVICE DEPLOYMENT

Updates to infrastructure during RP2

The OpenRiskNet VE requires a virtual infrastructure to be instantiated on either a public cloud (IaaS) provider or a local computer resource. A great deal of work has gone into supporting a cloud-agnostic infrastructure. The resulting VE has been tested on Google Cloud, Amazon Web Services, and on OpenStack.

Automating deployment

Deployment was initially described in D2.3 and has seen substantial updates during RP2. The project has shifted to use OKD Orchestrator (https://github.com/InformaticsMatters/okd-orchestrator) as the preferred tool for the formation of the OpenRiskNet VE deployments. The role of the OKD Orchestrator is to provide a simplified installation process that supports these types of deployments:

- simple one-server deployment for basic experimentation;
- standard availability allowing moderate scalability;
- high availability providing a high level of fault tolerance and scalability.

We initially aimed to support these infrastructures:

- OpenStack - for flexible use and deployment to in-house clusters
- Amazon AWS - for robust cloud production deployments
- Google CE - for robust cloud production deployments
- Bare Metal - for custom/on-premise infrastructure
- Scaleway - for low cost cloud deployments

Of these, OpenStack is the most tested option as the Swedish Science Cloud (SSC) reference and development sites use it. Bare metal (e.g. physical servers with no virtualisation) is also well tested as the JGU reference site (that replaced the one on SSC) and the VE operated by Diamond Light Source run on machines (physical machines in the case of Diamond, VMs in the case of the JGU site) provisioned by the infrastructure provider.

We have performed ‘proof of principle’ deployments to AWS and GCE using the OKD Orchestrator. In contrast, we decided not to investigate Scaleway further as, although it provides a very cheap cloud hosting environment, it still does not provide all the necessary features.

The OKD Orchestrator is documented on https://docs.informaticsmatters.com.

OpenRiskNet Environments

During this reporting period the OpenRiskNet environments underwent significant reorganisation.

Reference site

This site (https://prod.openrisknet.org/), sometimes alternatively referred to as the ‘production’ site, was initially running on the Uppmax region of the SSC. During early 2019, the OpenShift version was changed from 3.7 to 3.11. As this involved skipping several versions, we decided to create a completely new cluster for this still running on the Uppmax region of the SSC instead of upgrading the old one. For the new site we chose not to deploy GlusterFS as it had proved problematic and instead used NFS volumes which require manual setup but have proved to be more reliable.

The automation of the deployment (using OKD Orchestrator) made this a relatively straight forward process, following which the partner applications were redeployed. Again, because of the efforts to streamline deployments, this was a relatively straight forward process.

The environment at Uppmax consisted of:

- 1 bastion node with 2 CPU cores and 4GB RAM
- 1 master node with 8 CPU cores and 16GB RAM
- 1 infrastructure node with 8 CPU cores and 16GB RAM
- 10 worker nodes with 8 CPU cores and 16GB RAM
- Storage supplied as OpenStack Cinder volumes

Towards the end of the project, due to the uncertainty of retaining resources at SSC and because of issues with SSC resource provisioning, we looked for alternative options. The BEAR cloud at UOB was considered, but security restrictions prevented that being viable, and JGU offered access to servers at their site. Thus, the reference site was moved to servers at JGU in early November 2019. This comprises:

- 1 bastion node with 4 CPU cores and 8GB RAM
- 1 master node with 8 CPU cores and 32GB RAM
- 1 infrastructure node with 8 CPU cores and 32GB RAM
- 3 worker nodes with 60 CPU cores and 96GB RAM
- Storage as attached physical volumes, exposed as NFS volumes

Although there are fewer machines, they are individually more powerful and the overall compute performance is higher than on Uppmax.

The majority of the work to relocate to JGU took around 2 days and included:

- Transfer of data from the core PostgreSQL database, including
  - List of users in the Single Sign-On (SSO) system
  - Event logs from SSO (e.g. User login data)
  - Application data (e.g. Squonk, Modelling Web)
- Relocation of Jupyter notebook data of users
- Redeployment of partner and associated partner applications

JGU have committed to provide access to this infrastructure for a period of at least 2 years and to perform routine maintenance and patching to the servers and VMs. This provides a significant period during which the project’s outputs can be sustained once the project finishes. JGU should be recognised for providing this to the project free of charge, as should the SSC for providing facilities during the project.

Development sites

The OpenRiskNet development site (available at https://dev.openrisknet.org:8443/) is running on the HPC2N region of the SSC and is used by project members for testing and development purposes.

Like the reference site, it was running OpenShift 3.7 at the start of this reporting period and was upgraded to 3.11 during 2019. This upgrade was a precursor and test for upgrading the reference site.

We do not expect to need a development site once the project completes.

Other VEs

Some project partners have created additional VEs for their own use on different environments such as AWS and GCE. A description of these is out of scope for this demonstrator report and, thus, we will only exemplify here one such VE.

Of particular note is the VE that has been set up at the Diamond Light Source (an associated partner). IM has provided assistance in setting this up and managing it. The site operates on bare metal servers (OpenStack 'Ironic') running on the Verne Global cloud in Iceland. The VE is used to support Diamond's fragment screening follow up efforts, with the key application being their 'Fragalysis stack' which can be accessed at https://fragalysis.diamond.ac.uk. The site also includes deployment of some OpenRiskNet partner applications, such as the Squonk Computational Notebook. The OpenRiskNet VE concept was an excellent match to Diamond's needs, with the chemical risk assessment aspects of OpenRiskNet partner applications being an additional benefit.

Some pharmaceutical companies are now looking to deploy their own versions of 'Fragalysis stack', which should entail an OpenRiskNet VE (or the next iteration of the concept) being deployed within pharmaceutical companies during 2020.

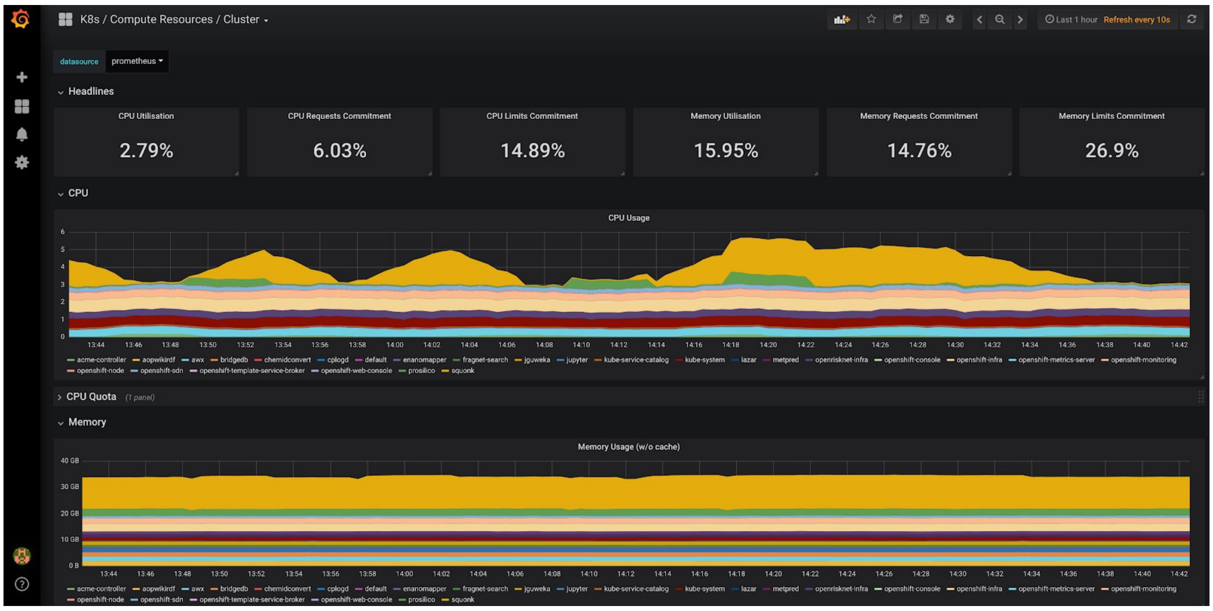
Monitoring

A basic setup of Prometheus for monitoring the VE was described in **Deliverable 2.3** [4] but little use had been made of this at this point of time.

The switch from OpenShift version 3.7 to 3.11 allowed a more functional deployment of Prometheus that can be used to monitor the OpenShift environment and to send alerts when certain metrics are outside their permitted ranges.

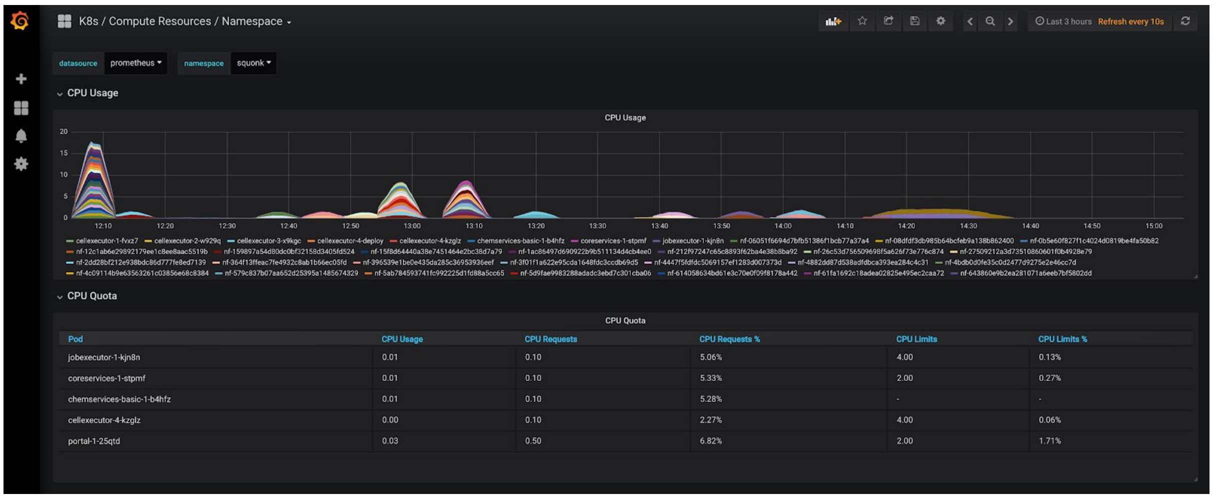
Prometheus can be used for monitoring the activity on the VE in a number of ways. One of the simplest is by a series of dashboards. Some come already deployed whilst others can be created for specific purposes. Here we will illustrate a couple of the generic dashboards.

The first provides an overview of the cluster activity:


K8s / Compute Resources / Cluster
datasource prometheus
Headlines
CPU Utilisation 2.79%
CPU Requests Commitment 6.03%
CPU Limits Commitment 14.89%
Memory Utilisation 15.95%
Memory Requests Commitment 14.76%
Memory Limits Commitment 26.9%
CPU
CPU Usage
CPU Quota (1 panel)
Memory
Memory Usage (w/o cache)

CPU and memory utilisation for each of the cluster nodes are shown, but various other information is also available.

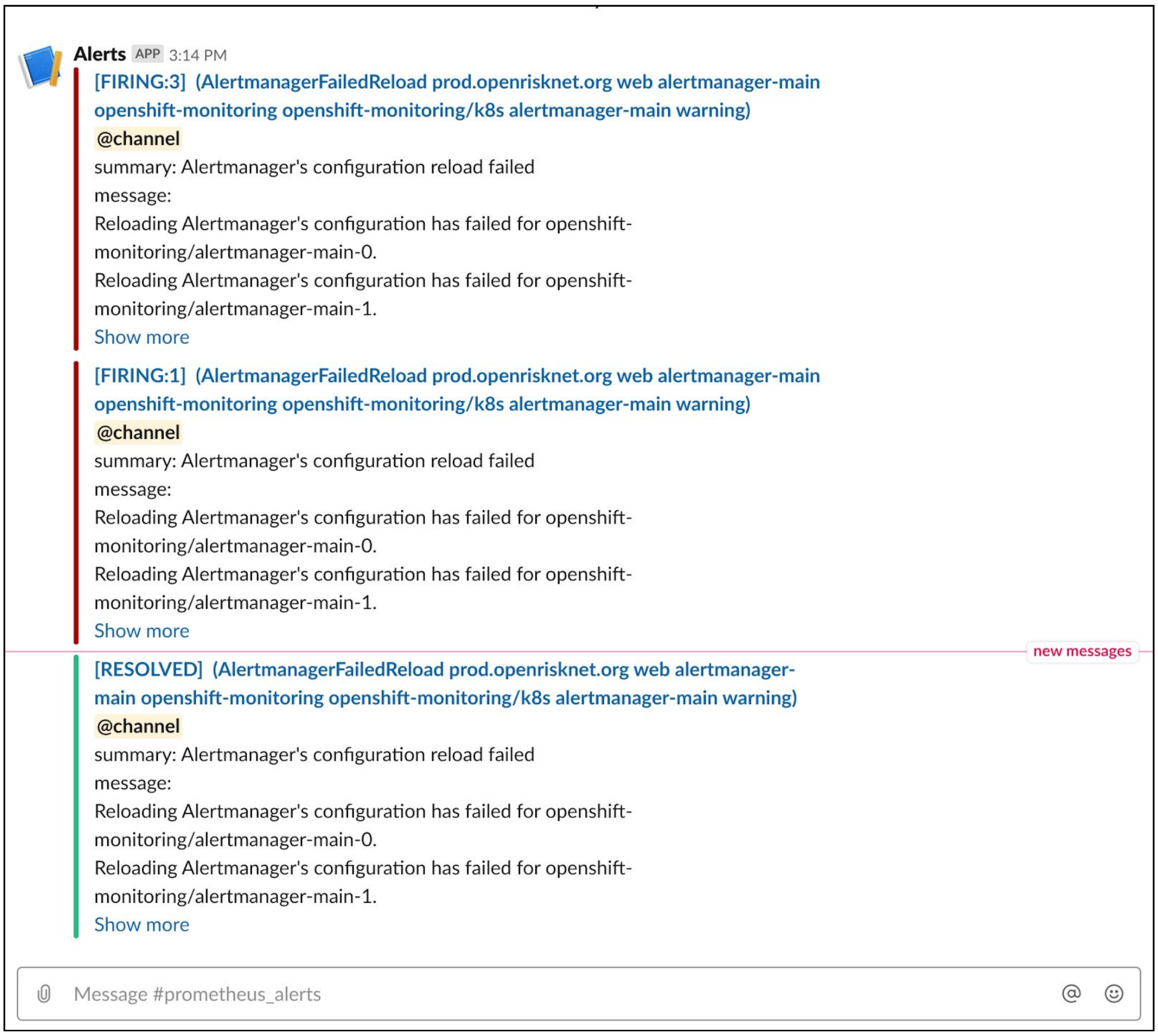
This second screenshot shows utilisation for each Kubernetes namespace.



CPU Quota

Pod	CPU Usage	CPU Requests	CPU Requests %	CPU Limits	CPU Limits %
jobexecutor-1-kjn8n	0.01	0.10	5.06%	4.00	0.13%
coreservices-1-stpmf	0.01	0.10	5.33%	2.00	0.27%
chemservices-basic-1-b4hfz	0.01	0.10	5.28%	-	-
cellexecutor-4-kzglz	0.00	0.10	2.27%	4.00	0.06%
portal-1-25qtd	0.03	0.50	6.82%	2.00	1.71%

In addition to providing a wide range of metrics, Prometheus allows generation of alerts when those metrics go outside the expected range. This is done through the Prometheus “Alert Manager”. We have set up basic alerting using the built in alerts for Kubernetes and direct these to a Slack channel used by the project. This way project members get notified if problems occur in the cluster. The following shows a series of alerts in the Slack channel:



Alerts APP 3:14 PM

```
[FIRING:3] (AlertmanagerFailedReload prod.openrisknet.org web alertmanager-main
openshift-monitoring openshift-monitoring/k8s alertmanager-main warning)
@channel
summary: Alertmanager's configuration reload failed
message:
Reloading Alertmanager's configuration has failed for openshift-
monitoring/alertmanager-main-0.
Reloading Alertmanager's configuration has failed for openshift-
monitoring/alertmanager-main-1.
Show more

[FIRING:1] (AlertmanagerFailedReload prod.openrisknet.org web alertmanager-main
openshift-monitoring openshift-monitoring/k8s alertmanager-main warning)
@channel
summary: Alertmanager's configuration reload failed
message:
Reloading Alertmanager's configuration has failed for openshift-
monitoring/alertmanager-main-0.
Reloading Alertmanager's configuration has failed for openshift-
monitoring/alertmanager-main-1.
Show more

new messages

[RESOLVED] (AlertmanagerFailedReload prod.openrisknet.org web alertmanager-
main openshift-monitoring openshift-monitoring/k8s alertmanager-main warning)
@channel
summary: Alertmanager's configuration reload failed
message:
Reloading Alertmanager's configuration has failed for openshift-
monitoring/alertmanager-main-0.
Reloading Alertmanager's configuration has failed for openshift-
monitoring/alertmanager-main-1.
Show more
```

At this stage, our usage of metrics and alerts is relatively basic, but the mechanism for making this more sophisticated is completely in place.

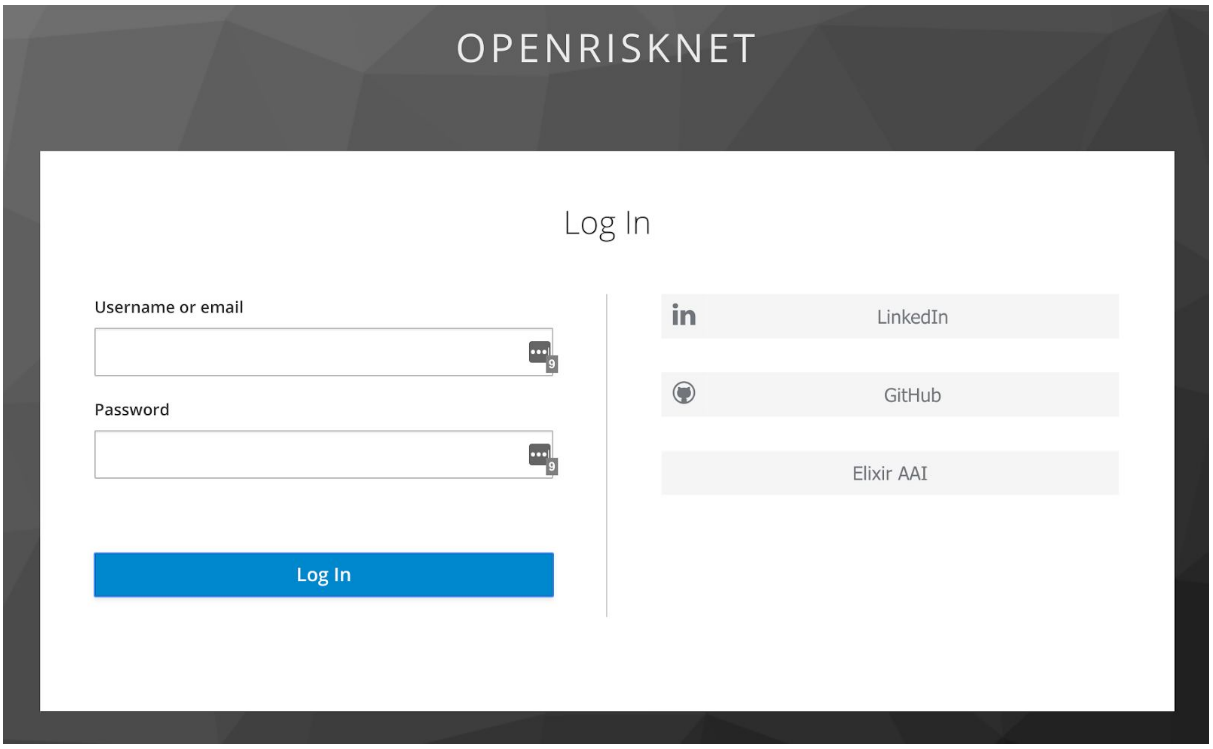
Access to the Prometheus systems requires administrative privileges and is not available to end users.

Security environment

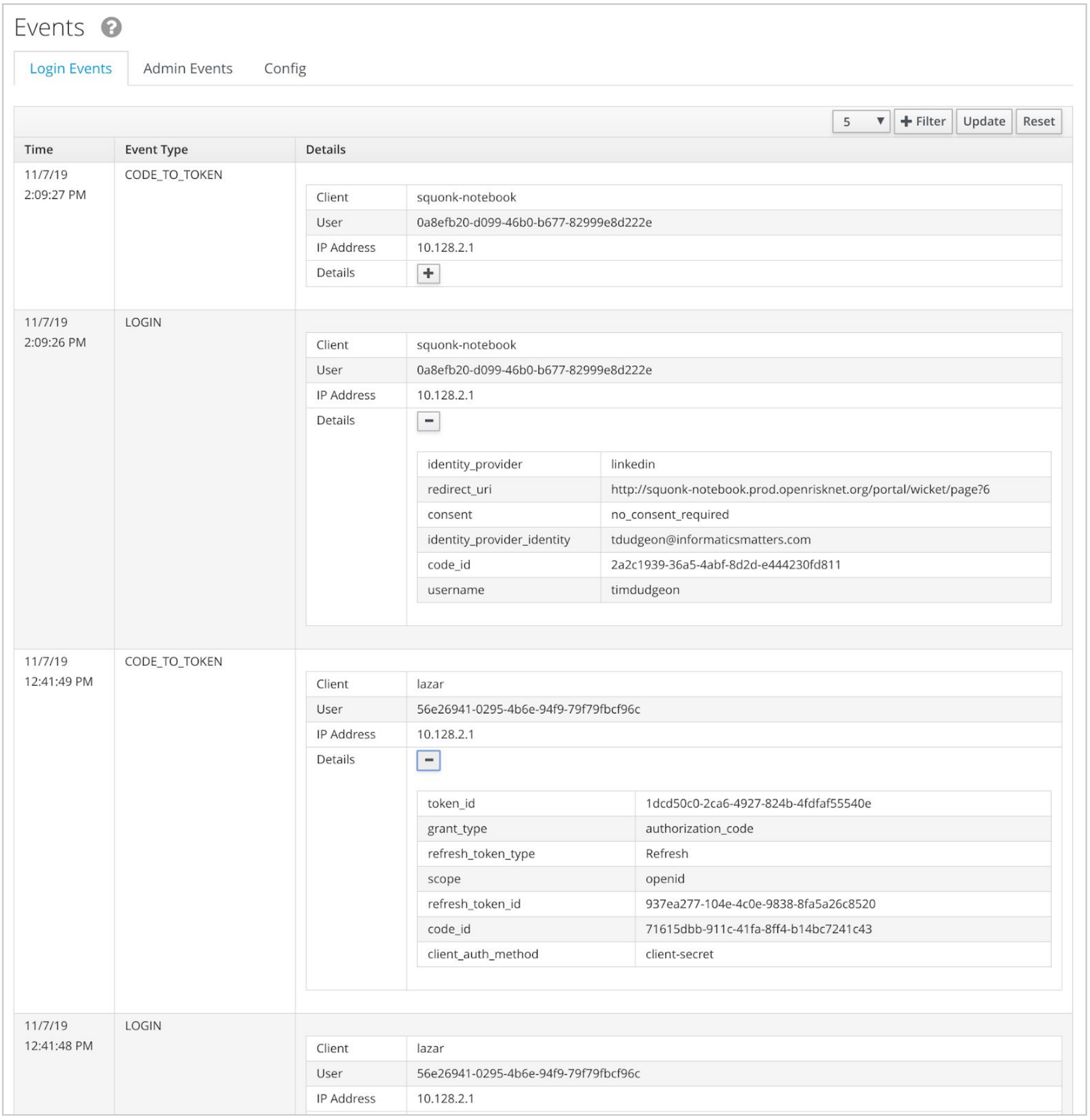
This was initially described in D2.3 and has during RP2 been updated and maintained. The main activity was upgrading the Keycloak (SSO server) to version 4.8.3.Final, which was needed because of a change to the APIs used by the LinkedIn Identity Provider. As part of this upgrade process, the deployment process was streamlined making it easier to upgrade in the future. All data in the SSO server was retained during the upgrade. These procedures can be found at https://github.com/OpenRiskNet/home/tree/master/openshift/deployments/openrisknet-infra.

As an addition to LinkedIn and GitHub, the authentication mechanisms used by Elixir (AAI) was added as an Identity Provider. This allows Elixir user to access the OpenRiskNet environment using their Elixir credentials. The complete set of options for logging in to

OpenRiskNet is shown below, the user accounts stored in KeyCloak (typically administrators) can be used on the left hand side, and any of the Identity Providers can be used on the right hand side.



Keycloak provides a detailed event log allowing us to monitor, for instance, who logs in to access services.



Time	Event Type	Details
11/7/19 2:09:27 PM	CODE_TO_TOKEN	Client: squonk-notebook User: 0a8efb20-d099-46b0-b677-82999e8d222e IP Address: 10.128.2.1 Details: +
11/7/19 2:09:26 PM	LOGIN	Client: squonk-notebook User: 0a8efb20-d099-46b0-b677-82999e8d222e IP Address: 10.128.2.1 Details: − identity_provider: linkedin redirect_uri: http://squonk-notebook.prod.openrisknet.org/portal/wicket/page?6 consent: no_consent_required identity_provider_identity: tdudgeon@informaticsmatters.com code_id: 2a2c1939-36a5-4abf-8d2d-e444230fd811 username: timdudgeon
11/7/19 12:41:49 PM	CODE_TO_TOKEN	Client: lazar User: 56e26941-0295-4b6e-94f9-79f79fbcf96c IP Address: 10.128.2.1 Details: − token_id: 1dcd50c0-2ca6-4927-824b-4fdfaf55540e grant_type: authorization_code refresh_token_type: Refresh scope: openid refresh_token_id: 937ea277-104e-4c0e-9838-8fa5a26c8520 code_id: 71615dbb-911c-41fa-8ff4-b14bc7241c43 client_auth_method: client-secret
11/7/19 12:41:48 PM	LOGIN	Client: lazar User: 56e26941-0295-4b6e-94f9-79f79fbcf96c IP Address: 10.128.2.1

Service discovery and registry

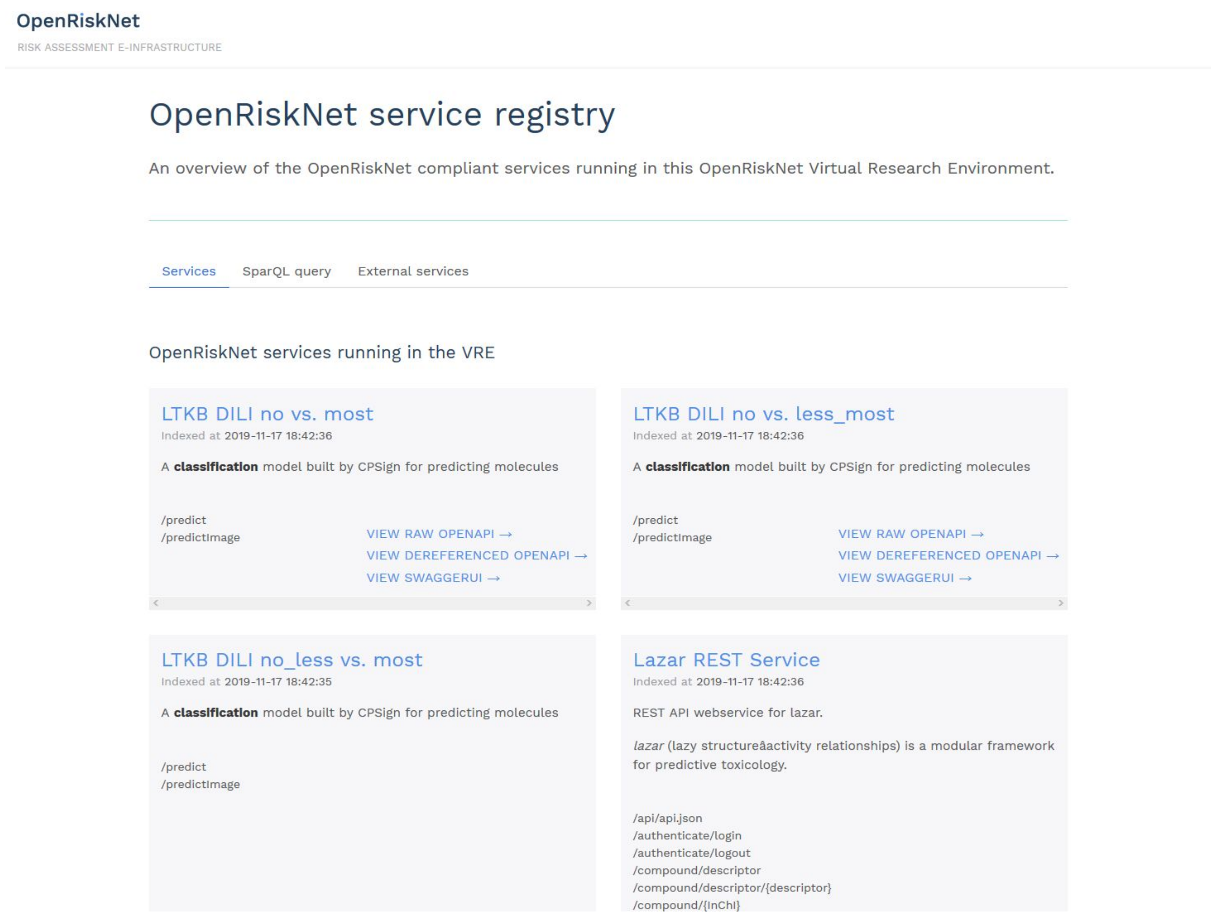
This was initially described in D2.3 [4] and again in detail in D2.4 [3]. During RP2, work has continued to improve the OpenRiskNet service registry. The source code is available at https://github.com/openrisknet/registry. The most important improvements since D2.4 are listed below.

One important enhancement was the ability to add external service definitions - this enables the service registry to parse OpenRiskNet annotated OpenAPI service definitions, not just of services running inside the same Virtual Research Environment (VRE), but also of any other service deployed in the public Internet or other accessible network. This allows services running on the public and that are not intended to be deployed inside a VRE (e.g. because they come with large amounts of data that would be unfeasibly to deploy into a VRE or that can't be copied in such a form for legal reasons) to still be indexed and findable via the service registry using SparQL queries that target the semantic annotations described in detail in D2.4. Examples of such services include e.g. ToxPlanet and EdelweissData.

To guard access to the service registry and to be able to monitor for more informative usage statistics, the service registry was extended with Keycloak integration (the OpenRiskNet single sign-on solution) for authentication and authorization. By utilizing Keycloak via the OpenID Connect standard all the advanced features like federated logins using Github and LInkedIn accounts are available for the service registry.

Automatic refresh was added to the registry so that services that generate their OpenRiskNet-annotated OpenAPI definitions dynamically (i.e. changing at runtime) get periodically refreshed. This allows services to update the semantic annotations at runtime (e.g. if new datasets or models are added as endpoints on an existing service).

In anticipation of a growing number of services indexed by the service registry, several performance tweaks were made, e.g. SparQL queries are now evaluated in parallel on all services.


OpenRiskNet
RISK ASSESSMENT E-INFRASTRUCTURE

OpenRiskNet service registry

An overview of the OpenRiskNet compliant services running in this OpenRiskNet Virtual Research Environment.

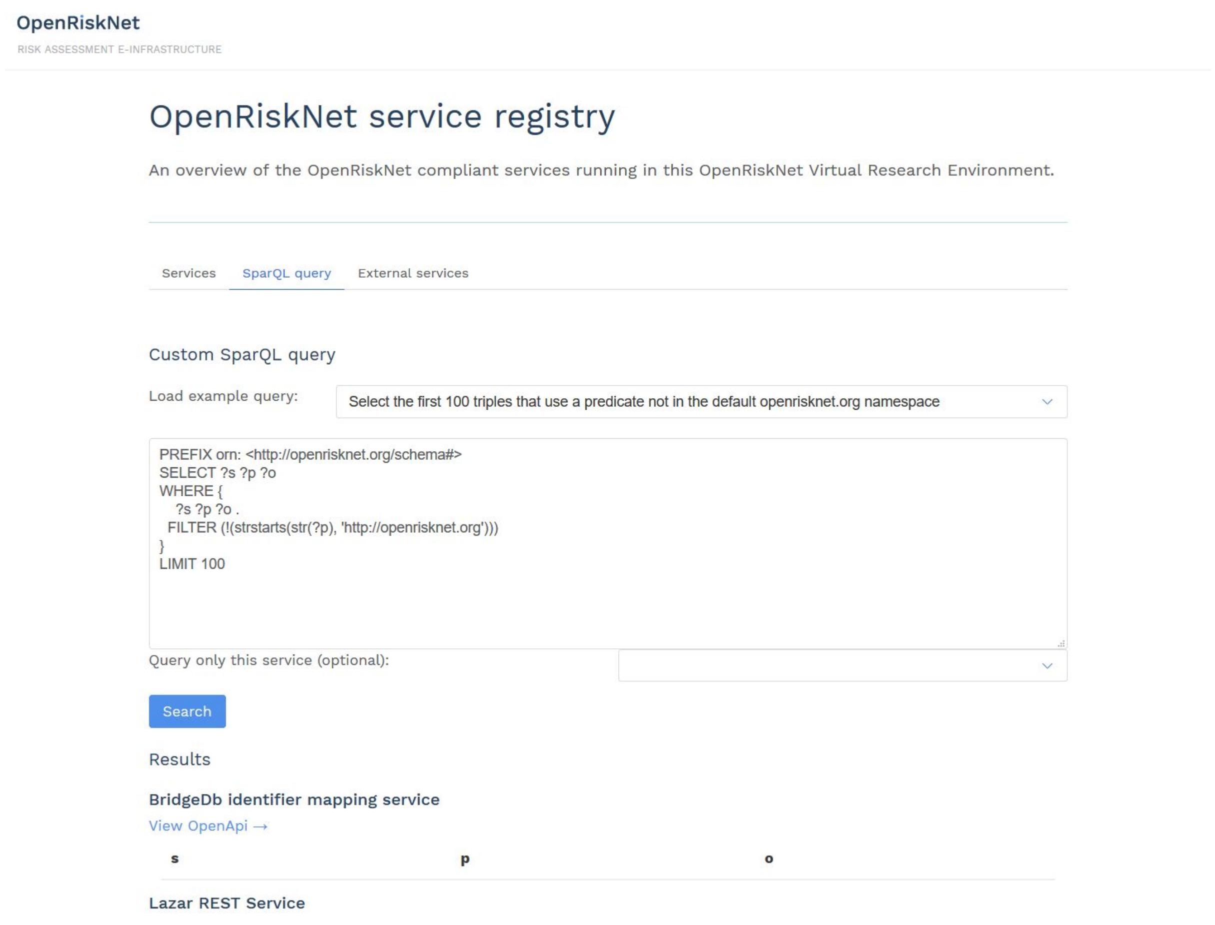
OpenRiskNet services running in the VRE

LTKB DILI no vs. most
Indexed at 2019-11-17 18:42:36
A **classification** model built by CPSign for predicting molecules
/predict
/predictImage
VIEW RAW OPENAPI →
VIEW DEREFERENCED OPENAPI →
VIEW SWAGGERUI →

LTKB DILI no vs. less_most
Indexed at 2019-11-17 18:42:36
A **classification** model built by CPSign for predicting molecules
/predict
/predictImage
VIEW RAW OPENAPI →
VIEW DEREFERENCED OPENAPI →
VIEW SWAGGERUI →

LTKB DILI no_less vs. most
Indexed at 2019-11-17 18:42:35
A **classification** model built by CPSign for predicting molecules
/predict
/predictImage

Lazar REST Service
Indexed at 2019-11-17 18:42:36
REST API webservice for lazar.
lazar (lazy structureâactivity relationships) is a modular framework for predictive toxicology.
/api/api.json
/authenticate/login
/authenticate/logout
/compound/descriptor
/compound/descriptor/{descriptor}
/compound/{InChI}



OpenRiskNet service registry

An overview of the OpenRiskNet compliant services running in this OpenRiskNet Virtual Research Environment.

Custom SparQL query

Load example query: Select the first 100 triples that use a predicate not in the default openrisknet.org namespace

```
PREFIX orn: <http://openrisknet.org/schema#>
SELECT ?s ?p ?o
WHERE {
   ?s ?p ?o .
  FILTER (!(strstarts(str(?p), 'http://openrisknet.org')))
}
LIMIT 100
```

Query only this service (optional):

Results

BridgeDb identifier mapping service

View OpenApi →

s	p	o

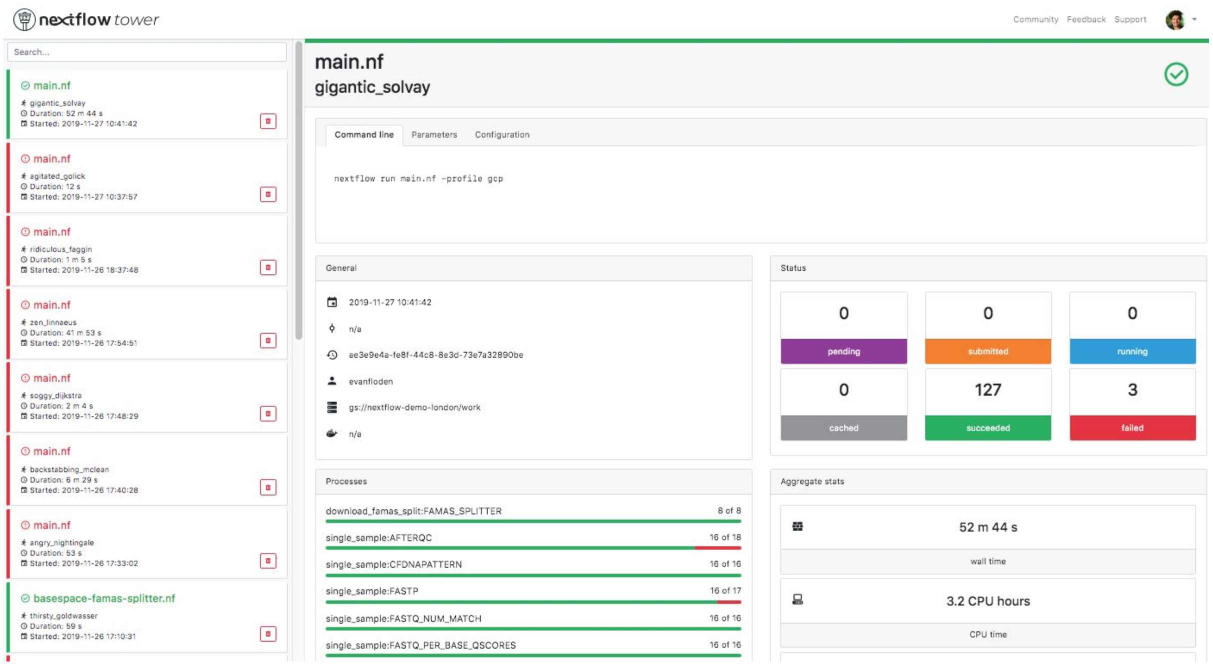
Lazar REST Service

Interconnecting with external infrastructures

This was initially described in D2.5 [5] and during RP2, the updates have been related to the ease-of-use improvements related to external infrastructure and how these can connect with the reference VE. The launching of workloads into external infrastructure is performed using the Nextflow workflow manager, which is a single-user, single-execution, command-line application. For this improvement, we wanted target workloads that could be launched externally, for example from institutional computing cluster, and monitored from a centralised location hosted on the VE. This situation is common where large datasets are located within one data centre and the resource intensive computation must be performed in that location.

To achieve this goal of interconnecting with external infrastructures, we deployed Nextflow Tower into the reference VE. Nextflow Tower is an open source monitoring and managing platform for Nextflow workflows. It consists of a front-end service for the monitoring of workflows that are launched from any infrastructure that has web access. An API connects single Nextflow instances (workflow executions) to the centralised service within the VE. A MySQL backend-database stores a complete history of all the workflow executions. Information relating to each task, including requested and utilised memory, cpu and time are recorded along with container information and efficiency.

Nextflow Tower users are authenticated via email address with a whitelist function available to allow the pre-authorisation of organisational email domains. Users log in through the login page and are greeted with a set of instructions on how they can deploy workflows that can be monitored from the VE. It should be stressed that any existing Nextflow workflow and execution environment is compatible with Nextflow Tower.



main.nf
gigantic_solvay
Duration: 52 m 44 s
Started: 2019-11-27 10:41:42

main.nf
agitated_golick
Duration: 12 s
Started: 2019-11-27 10:37:57

main.nf
ridiculous_faggin
Duration: 1 m 5 s
Started: 2019-11-26 18:37:48

main.nf
zen_linnaeus
Duration: 41 m 53 s
Started: 2019-11-26 17:54:51

main.nf
soggy_dijkstra
Duration: 2 m 4 s
Started: 2019-11-26 17:48:29

main.nf
backstabbing_mclean
Duration: 6 m 29 s
Started: 2019-11-26 17:40:28

main.nf
angry_nightingale
Duration: 53 s
Started: 2019-11-26 17:33:02

basespace-famas-splitter.nf
thirsty_goldwasser
Duration: 59 s
Started: 2019-11-26 17:10:31

main.nf
gigantic_solvay

```
nextflow run main.nf -profile gcp
```

General
2019-11-27 10:41:42
n/a
ae3e9e4a-fe8f-44c8-8e3d-73e7a32890be
evanfloden
gs://nextflow-demo-london/work
n/a

Status

0	0	0
pending	submitted	running
0	127	3
cached	succeeded	failed

Processes

download_famas_split:FAMAS_SPLITTER	8 of 8
single_sample:AFTERQC	16 of 18
single_sample:CFDNAPATTERN	16 of 16
single_sample:FASTP	16 of 17
single_sample:FASTQ_NUM_MATCH	16 of 16
single_sample:FASTQ_PER_BASE_QSCORES	16 of 16

Aggregate stats
52 m 44 s
wall time
3.2 CPU hours
CPU time

The application has been designed to be deployed as 2 microservices (web and backend). This makes it particularly appealing for deployment within the OpenShift VE, which is designed for containerised installation and management of services.

Demonstrator

The Demonstrator, in the form of the OpenRiskNet reference installation, is available on https://home.prod.openrisknet.org/. The page provides links to the systems in the reference site. Administrator rights will be needed to access some of these systems. Basic usage instructions can be accessed from that page. Until 26th November 2019, there have been 104 users of the reference site according to the statistics of the single-sign-on (SSO, see above). This number should be seen as a lower bound since some of the tools don't need login into the SSO and, therefore, cannot be monitored this way. Another way of monitoring the usage is the number of accesses of the starting page of the reference instances. This was monitored by Google Analytics and gave 182 page views in the period from 1 June to 26 November 2019. More details on the website accesses can be found in Deliverable D3.5.

Documentation

The documentation for the project is primarily in GitHub projects owned by the OpenRiskNet organisation (https://github.com/OpenRiskNet), most importantly the 'home' repo which can be found here: https://github.com/OpenRiskNet/home. The OpenRiskNet Wiki (https://github.com/OpenRiskNet/home/wiki) also contains documentation for the infrastructure. All these documents are publicly accessible under open licenses.
Of particular note are:

- Recipes - a set of 'howto' instructions for common tasks
- Deployments - a set of materials and instructions for deploying OpenRiskNet partner applications
- Environments - a set of materials and instructions for provisioning an OpenRiskNet VE
- OKD Orchestrator - the tooling used to deploy OpenRiskNet VEs
- Deployment Guidelines - best practices for application deployment to a VE
- Notebooks - collection of notebooks for consuming OpenRiskNet services

Example 1: Deploying the VE

Deploying a VE is a relatively complex process and needs administrative rights on the host computers. It is not something that is easily demonstrated in real time. Instead we refer to the recording of the webinar that was run to illustrate this process to the wider community, where we walk through this procedure of creating a VE:

https://www.youtube.com/watch?v=qOiOC09XRIg

Example 2: Deploying a service in the VE

Extensive documentation on how to deploy services in OpenRiskNet VE is available in the material created for the "Deploying applications" workshop held on 23 October 2019 in Amsterdam. The workshop material can be found here:
https://github.com/OpenRiskNet/workshop/tree/master/wp2-deployment-workshop-2019

The topics covered were:

- Setup - Instructions for getting started
- Tutorial 1 - Introduction to containers, Kubernetes and OpenShift
- Exercise A - Deploying the PySimple container using the web console
- Tutorial 2 - Description of key Kubernetes objects
- Exercise B - Deploying the PySimple container using the CLI
- Exercise C - Deploying Lazar - a real world application (a video tutorial for this exercise is available (see below))
- Tutorial 3 - Introduction to persistent storage
- Exercise D - Deploying a persistent PySimple
- Tutorial 4 - Configuration
- Tutorial 5 - Probes and resource limits
- Tutorial 6 - Advanced deployment techniques

That workshop was based on the material from a previous webinar that was run for the wider community with the purpose of illustrating application deployment to a VE. A recording of the webinar is available on OpenRiskNet playlist in YouTube.

OpenRiskNet Webinar, 2019

OpenRiskNet

RISK ASSESSMENT E-INFRASTRUCTURE

Deploying Applications to an OpenRiskNet Virtual Environment

by

Tim Dudgeon and Alan Christie (Informatics Matters Ltd, UK)

Deploying OpenRiskNet applications tutorial https://youtu.be/qLgxaTPiKNc

OpenRiskNet

RISK ASSESSMENT E-INFRASTRUCTURE

Deployment of a a real-world OpenRiskNet application (Lazar) from the OpenShift Command-Line

by

Alan Christie

(Informatics Matters Ltd., UK)

Workshop Exercise C tutorial https://youtu.be/llYIzdgbv7o

Example 3: Logging in to the VE

Several of the partner applications deployed to the reference environment require a login to access. This is done through the SSO server (Keycloak) that is deployed as part of the infrastructure.

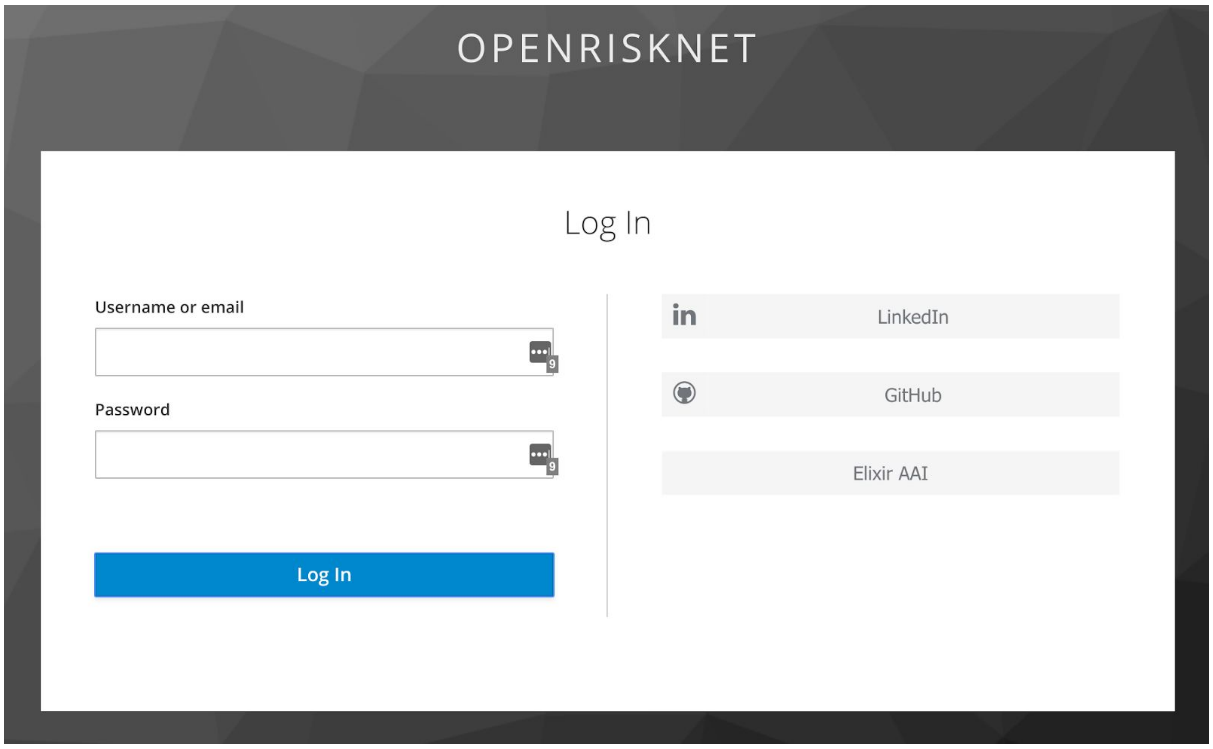
To demonstrate this we illustrate how to login and access two applications, The Squonk Computational Notebook and Jupyter Hub.

Step 1. Access the first application.

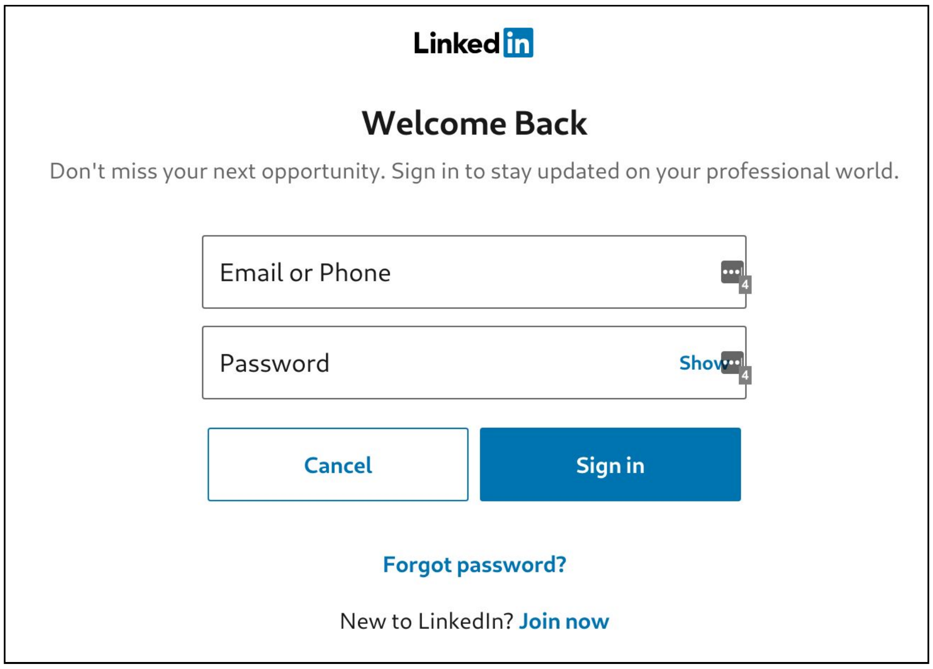
Go to https://squonk-notebook.prod.openrisknet.org/portal to access Squonk. As you will not yet be authenticated you are redirected to the SSO server to log in.

Step 2. Login to SSO

You will see a login page like this:



Choose one of the Identity Providers on the right hand side, e.g. LinkedIn, and you will be taken to the requested site and asked to login. In case of using LinkedIn, you will see something like this:

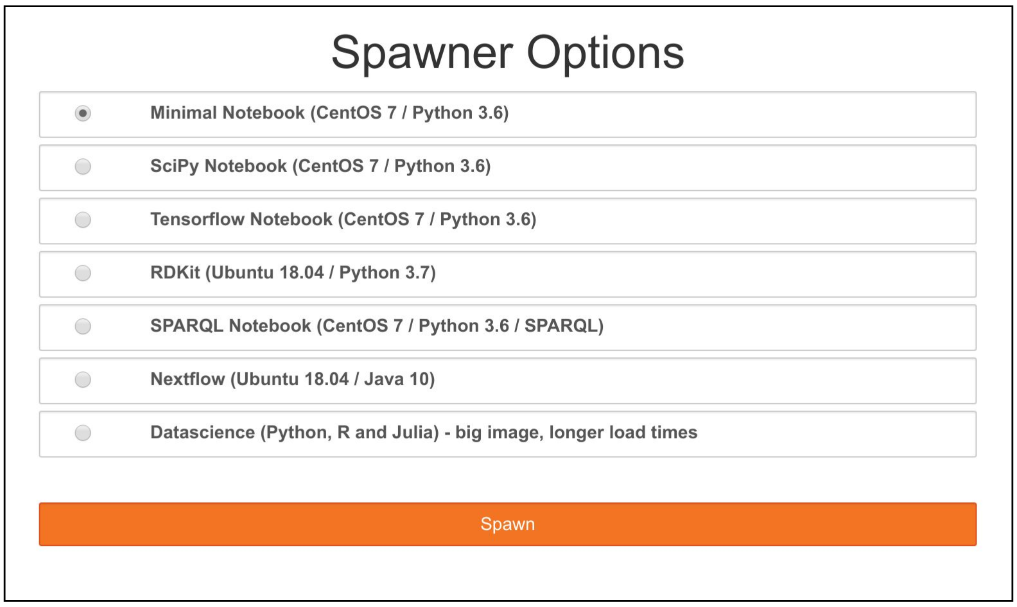

LinkedIn
Welcome Back
Don't miss your next opportunity. Sign in to stay updated on your professional world.
Email or Phone
Password
Show
Cancel
Sign in
Forgot password?
New to LinkedIn? Join now

Enter your username and password for that site (OpenRiskNet never sees your password) and if successful you are redirected back to Squonk Computational Notebook running in the OpenRiskNet reference environment. The LinkedIn site may ask for your permission to grant OpenRiskNet access to your details like your name and email address.

Step 3. Accessing JupyterHub

Now try to access JupyterHub and run a Jupyter Notebook. To do this go here: https://jupyter.prod.openrisknet.org/

As you are already logged in to SSO you can access this immediately and you should see a screen like this:


Spawner Options

[● Minimal Notebook (CentOS 7 / Python 3.6)]
SciPy Notebook (CentOS 7 / Python 3.6)
Tensorflow Notebook (CentOS 7 / Python 3.6)
RDKit (Ubuntu 18.04 / Python 3.7)
SPARQL Notebook (CentOS 7 / Python 3.6 / SPARQL)
Nextflow (Ubuntu 18.04 / Java 10)
Datascience (Python, R and Julia) - big image, longer load times

Choose one of the notebook types (the first one should be fastest) and a short while later you will have a Jupyter notebook session.

Alternative flow: Access Jupyter then Squonk

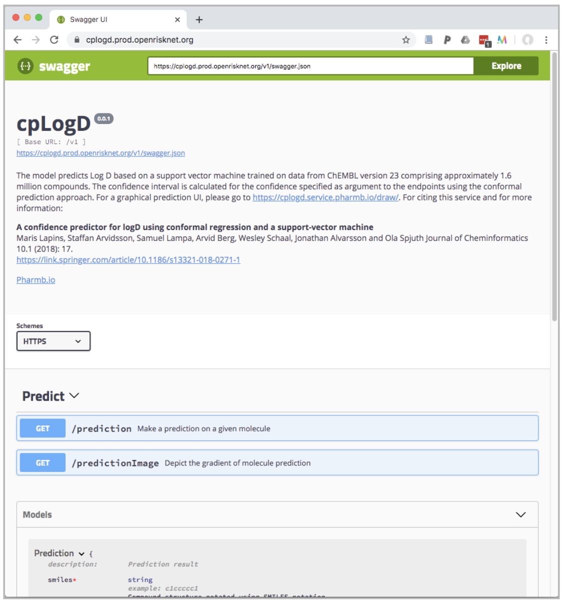
Had you chosen to access Jupyter first you would have been prompted to log in at that stage, and then would be able to access Squonk without having to log in again.

Example 4: Consuming a service

OpenRiskNet services can be consumed in different ways, including from its API directly, via specialized GUIs, programmatically via code e.g., in a Notebook.

Consuming services via OpenAPI

Services in OpenRiskNet are required to present a compatible API (as defined in Deliverable 2.4) that conforms to OpenAPI definition. This presents all services with a Swagger GUI that can be used to consume the individual service. Below, the Swagger/OpenAPI user interface (left), where services can be consumed directly, and the result of pasting in the SMILES for the drug Omeprazole and observing the result, in this case the prediction of LogD (service running at https://cplogd.prod.openrisknet.org/) and a rendering of individual atom's contribution to the prediction (right), are presented:


Swagger UI
cplogd.prod.openrisknet.org
swagger
https://cplogd.prod.openrisknet.org/v1/swagger.json
Explore
Schemes
HTTPS
Models

cpLogD 0.0.1
[Base URL: /v1]
https://cplogd.prod.openrisknet.org/v1/swagger.json

The model predicts Log D based on a support vector machine trained on data from ChEMBL version 23 comprising approximately 1.6 million compounds. The confidence interval is calculated for the confidence specified as argument to the endpoints using the conformal prediction approach. For a graphical prediction UI, please go to https://cplogd.service.pharmb.io/draw/. For citing this service and for more information:

A confidence predictor for logD using conformal regression and a support-vector machine
Maris Lapins, Staffan Arvidsson, Samuel Lampa, Arvid Berg, Wesley Schaal, Jonathan Alvarsson and Ola Spjuth Journal of Cheminformatics 10.1 (2018): 17.
https://link.springer.com/article/10.1186/s13321-018-0271-1

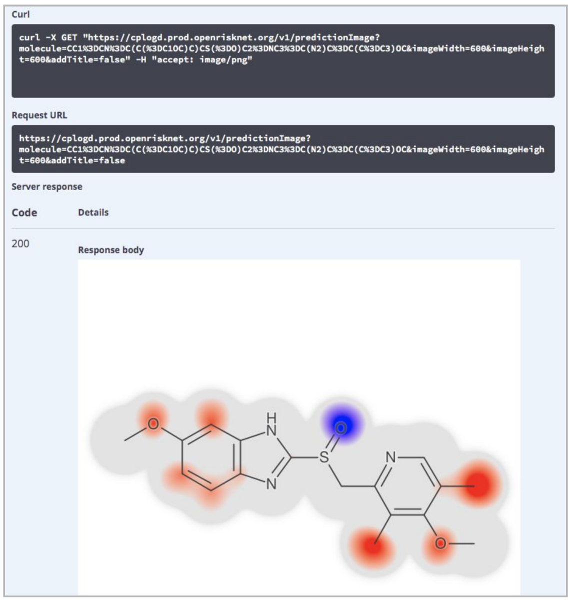
Pharmb.io

Predict

GET `/prediction` Make a prediction on a given molecule

GET `/predictionImage` Depict the gradient of molecule prediction

```
Prediction {
  description:    Prediction result
  smiles*         string
                  example: c1ccccc1
```



Curl

```
curl -X GET "https://cplogd.prod.openrisknet.org/v1/predictionImage?
molecule=CC1%3DCN%3DC(C(%3DC1OC)C)CS(%3DO)C2%3DNC3%3DC(N2)C%3DC(C%3DC3)OC&imageWidth=600&imageHeigh
t=600&addTitle=false" -H "accept: image/png"
```

Request URL

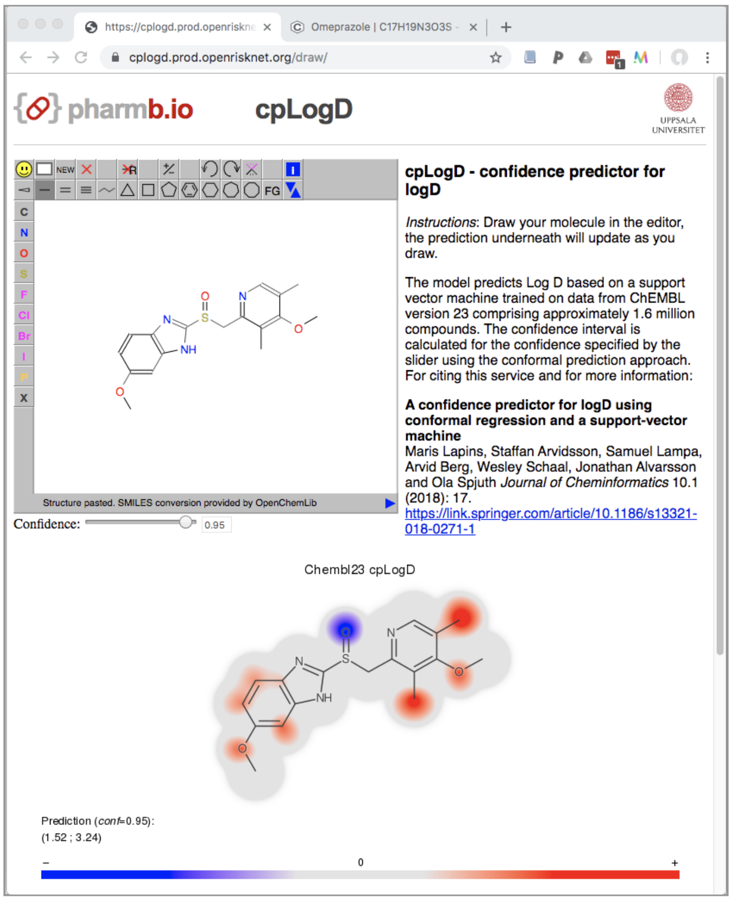
```
https://cplogd.prod.openrisknet.org/v1/predictionImage?
molecule=CC1%3DCN%3DC(C(%3DC1OC)C)CS(%3DO)C2%3DNC3%3DC(N2)C%3DC(C%3DC3)OC&imageWidth=600&imageHeigh
t=600&addTitle=false
```

Server response

Code	Details
200	Response body

Consuming services via GUI

If an OpenRiskNet service presents a GUI, this can also be used to consume the service. The figure below shows the same service (for predicting LogD) in a dedicated GUI, predicting LogD for the drug Omeprazole available at https://cplogd.prod.openrisknet.org/draw/):


https://cplogd.prod.openriskne
Omeprazole | C17H19N3O3S -
cplogd.prod.openrisknet.org/draw/
pharmb.io
cpLogD
UPPSALA UNIVERSITET
NEW
FG
C
N
O
S
F
Cl
Br
I
P
X
N
NH
O
S
N
O
O
N
S
N
NH
O
O
Chembl23 cpLogD
–
0
+

cpLogD - confidence predictor for logD

Instructions: Draw your molecule in the editor, the prediction underneath will update as you draw.

The model predicts Log D based on a support vector machine trained on data from ChEMBL version 23 comprising approximately 1.6 million compounds. The confidence interval is calculated for the confidence specified by the slider using the conformal prediction approach. For citing this service and for more information:

A confidence predictor for logD using conformal regression and a support-vector machine
Maris Lapins, Staffan Arvidsson, Samuel Lampa, Arvid Berg, Wesley Schaal, Jonathan Alvarsson and Ola Spjuth *Journal of Cheminformatics* 10.1 (2018): 17.
https://link.springer.com/article/10.1186/s13321-018-0271-1

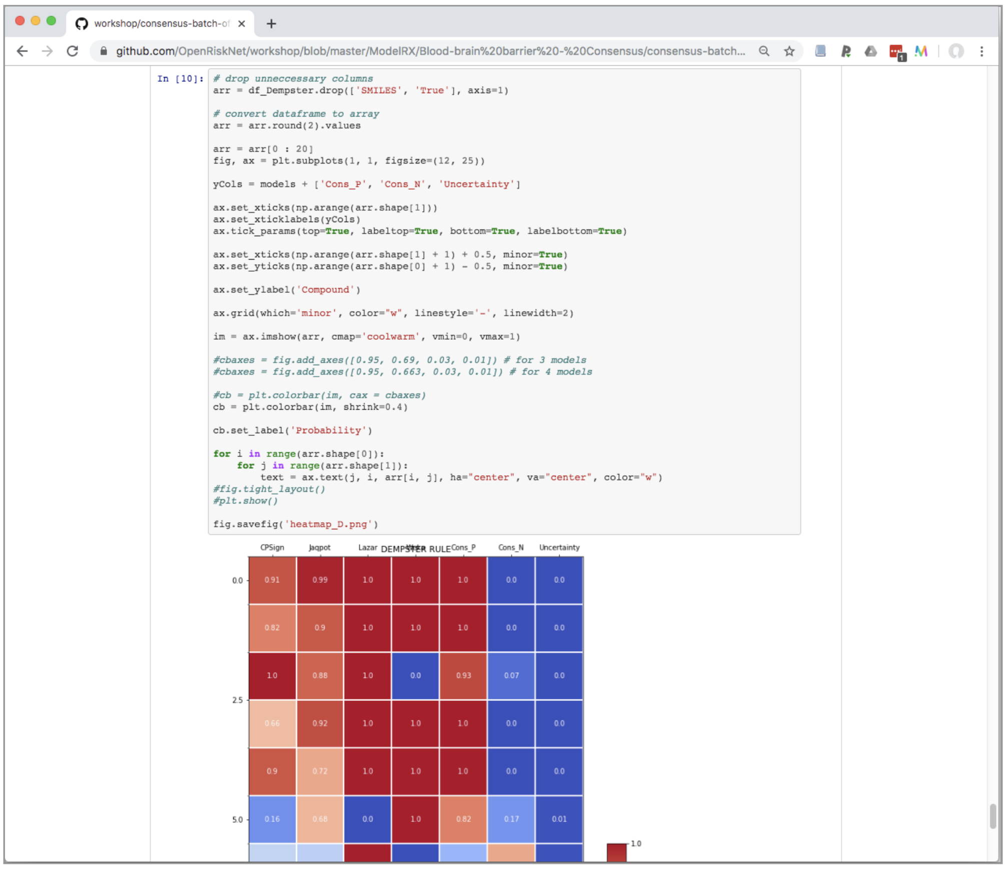
Structure pasted. SMILES conversion provided by OpenChemLib

Confidence: 0.95

Prediction (*conf*=0.95):
(1.52 ; 3.24)

Consuming services programmatically

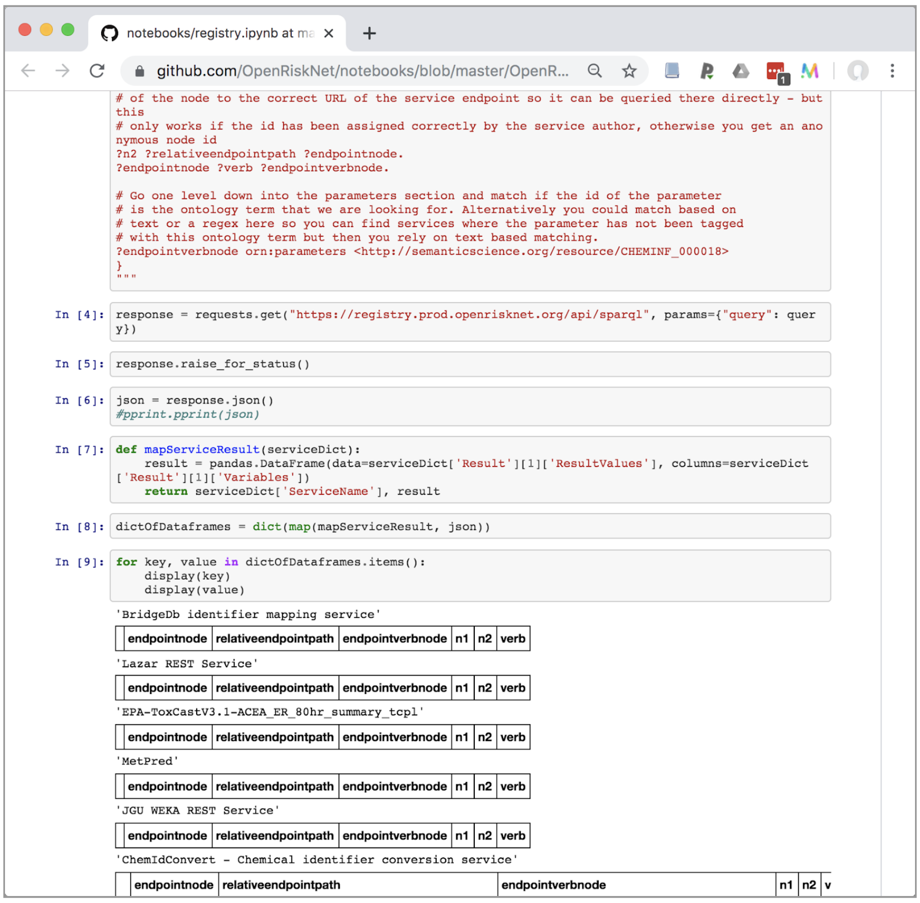
Services in OpenRiskNet can also be consumed programmatically via calling their API from code. A popular way of doing so is via Notebooks. OpenRiskNet has made available a number of different notebooks, available on https://github.com/OpenRiskNet/notebooks. The figure below shows an example notebook consuming multiple predictive modeling tools and performing a consensus modeling (notebook available at https://github.com/OpenRiskNet/workshop/tree/master/ModelRX/Blood-brain%20barrier%20-%20Consensus).


workshop/consensus-batch-of
github.com/OpenRiskNet/workshop/blob/master/ModelRX/Blood-brain%20barrier%20-%20Consensus/consensus-batch...

```
In [10]: # drop unneccessary columns
         arr = df_Dempster.drop(['SMILES', 'True'], axis=1)

         # convert dataframe to array
         arr = arr.round(2).values

         arr = arr[0 : 20]
         fig, ax = plt.subplots(1, 1, figsize=(12, 25))

         yCols = models + ['Cons_P', 'Cons_N', 'Uncertainty']

         ax.set_xticks(np.arange(arr.shape[1]))
         ax.set_xticklabels(yCols)
         ax.tick_params(top=True, labeltop=True, bottom=True, labelbottom=True)

         ax.set_xticks(np.arange(arr.shape[1] + 1) + 0.5, minor=True)
         ax.set_yticks(np.arange(arr.shape[0] + 1) - 0.5, minor=True)

         ax.set_ylabel('Compound')

         ax.grid(which='minor', color="w", linestyle='-', linewidth=2)

         im = ax.imshow(arr, cmap='coolwarm', vmin=0, vmax=1)

         #cbaxes = fig.add_axes([0.95, 0.69, 0.03, 0.01]) # for 3 models
         #cbaxes = fig.add_axes([0.95, 0.663, 0.03, 0.01]) # for 4 models

         #cb = plt.colorbar(im, cax = cbaxes)
         cb = plt.colorbar(im, shrink=0.4)

         cb.set_label('Probability')

         for i in range(arr.shape[0]):
             for j in range(arr.shape[1]):
                 text = ax.text(j, i, arr[i, j], ha="center", va="center", color="w")
         #fig.tight_layout()
         #plt.show()

         fig.savefig('heatmap_D.png')
```

Example 5: Locating services

A listing of the services available in the reference instance can be retrieved by querying the service registry, via a GUI available at http://registry.prod.openrisknet.org/ or programmatically. The figure below shows a notebook for querying the registry for services that accept a chemical structure in SMILES format as input (notebook available at https://github.com/OpenRiskNet/notebooks/blob/master/OpenRiskNetRegistry/registry.ipynb). See also section ‘Service discovery and Registry’ above.


notebooks/registry.ipynb at ma
github.com/OpenRiskNet/notebooks/blob/master/OpenR...

```
# of the node to the correct URL of the service endpoint so it can be queried there directly - but
this
# only works if the id has been assigned correctly by the service author, otherwise you get an ano
nymous node id
?n2 ?relativeendpointpath ?endpointnode.
?endpointnode ?verb ?endpointverbnode.

# Go one level down into the parameters section and match if the id of the parameter
# is the ontology term that we are looking for. Alternatively you could match based on
# text or a regex here so you can find services where the parameter has not been tagged
# with this ontology term but then you rely on text based matching.
?endpointverbnode orn:parameters <http://semanticscience.org/resource/CHEMINF_000018>
}
"""
```

```
In [4]: response = requests.get("https://registry.prod.openrisknet.org/api/sparql", params={"query": quer
        y})

In [5]: response.raise_for_status()

In [6]: json = response.json()
        #pprint.pprint(json)

In [7]: def mapServiceResult(serviceDict):
            result = pandas.DataFrame(data=serviceDict['Result'][1]['ResultValues'], columns=serviceDict
        ['Result'][1]['Variables'])
            return serviceDict['ServiceName'], result

In [8]: dictOfDataframes = dict(map(mapServiceResult, json))

In [9]: for key, value in dictOfDataframes.items():
            display(key)
            display(value)
```

'BridgeDb identifier mapping service'

	endpointnode	relativeendpointpath	endpointverbnode	n1	n2	verb

'Lazar REST Service'

	endpointnode	relativeendpointpath	endpointverbnode	n1	n2	verb

'EPA-ToxCastV3.1-ACEA_ER_80hr_summary_tcpl'

	endpointnode	relativeendpointpath	endpointverbnode	n1	n2	verb

'MetPred'

	endpointnode	relativeendpointpath	endpointverbnode	n1	n2	verb

'JGU WEKA REST Service'

	endpointnode	relativeendpointpath	endpointverbnode	n1	n2	verb

'ChemIdConvert - Chemical identifier conversion service'

	endpointnode	relativeendpointpath	endpointverbnode	n1	n2	v

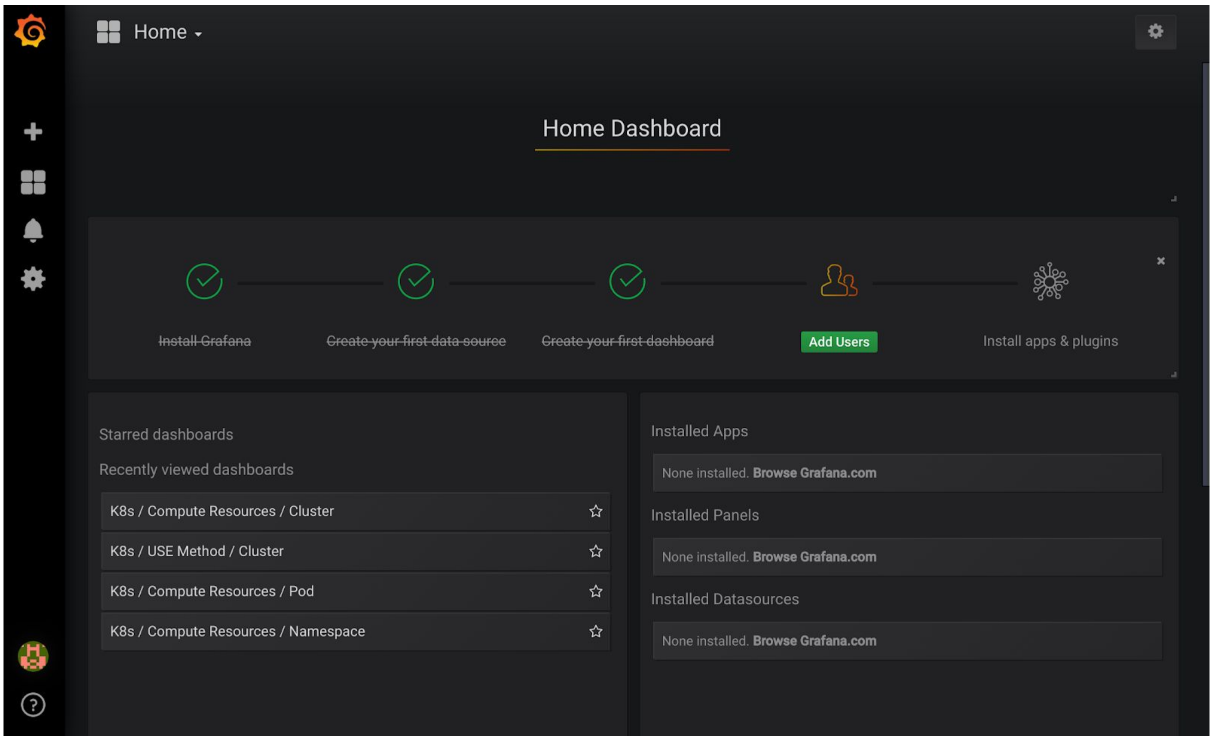
For further description on running and using applications, please refer to Deliverable D4.3.

Example 6: Monitoring the usage

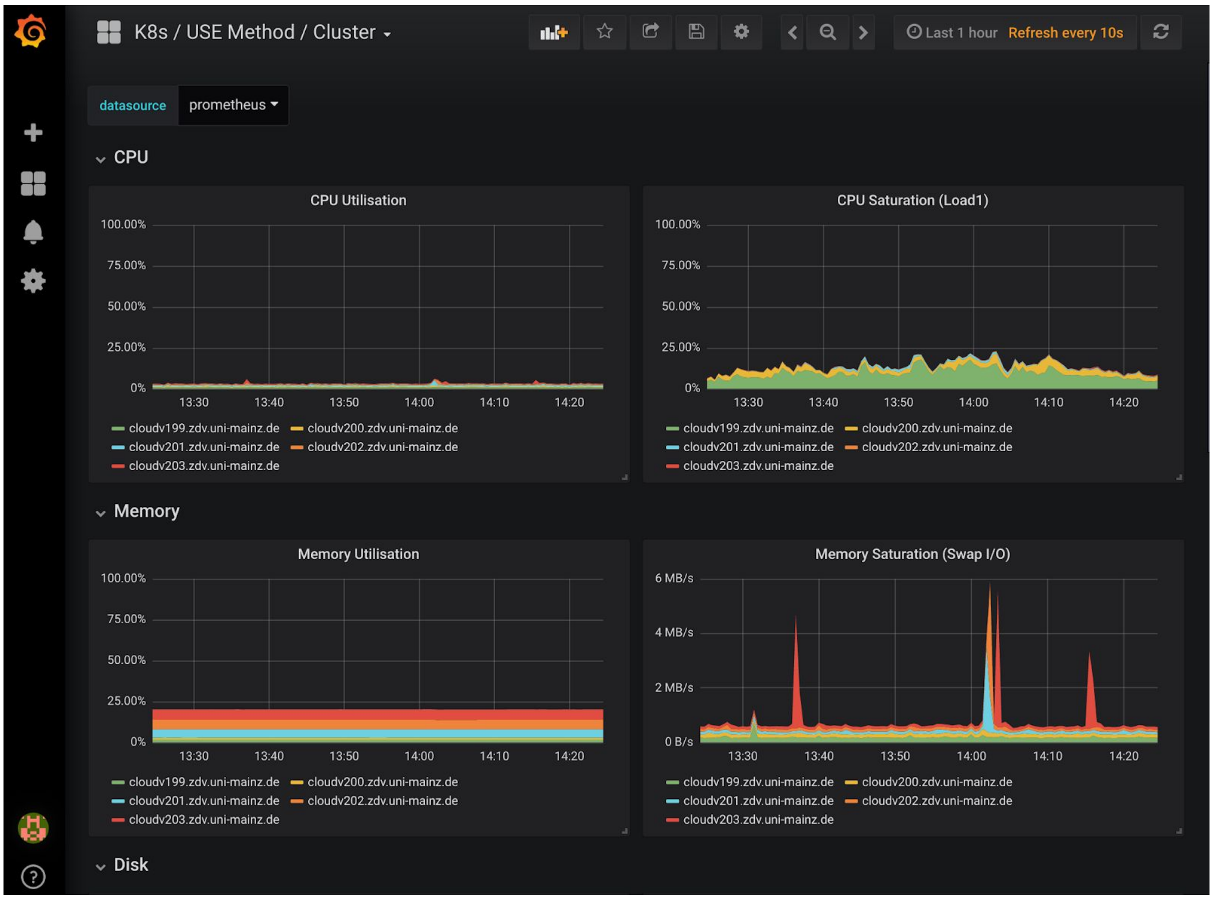
Metrics for the Kubernetes cluster are collected by Prometheus and stored in a time series database. They can be viewed either in Prometheus or using Grafana dashboards. Both require admin rights so we illustrate the process here in more detail since standard users cannot access these on the reference VE.

For viewing metrics using Grafana:

1. Connect to Grafana at https://grafana-openshift-monitoring.prod.openrisknet.org/
2. Login as one of the OpenRiskNet administrators
3. Select one of the available dashboards

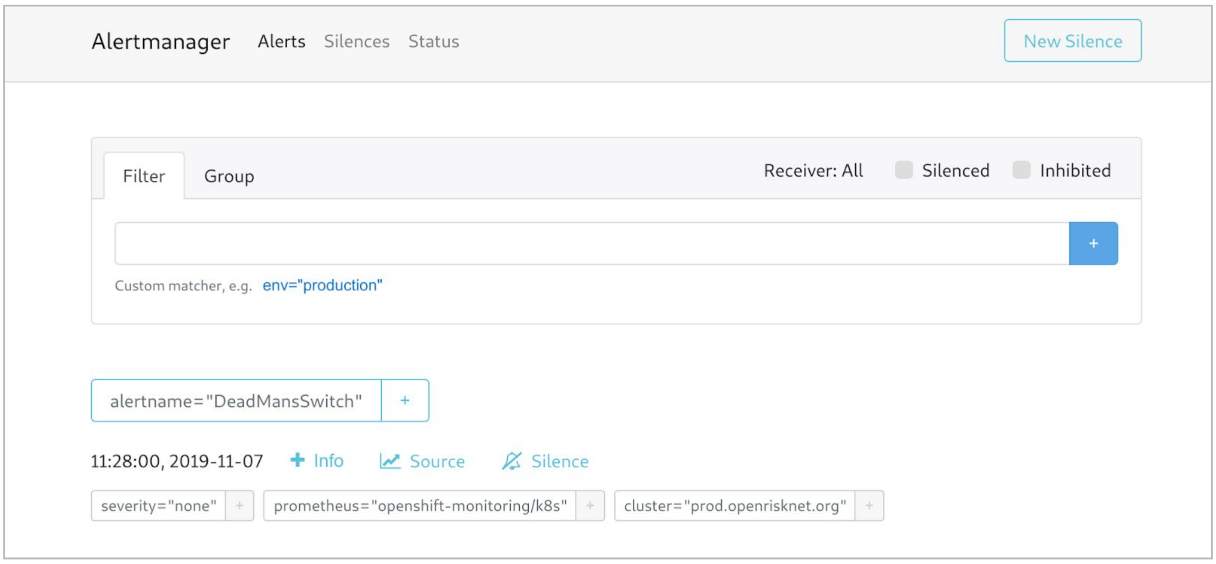

Home
Home Dashboard
~~Install Grafana~~
~~Create your first data source~~
~~Create your first dashboard~~
Add Users
Install apps & plugins
Starred dashboards
Recently viewed dashboards
K8s / Compute Resources / Cluster
K8s / USE Method / Cluster
K8s / Compute Resources / Pod
K8s / Compute Resources / Namespace
Installed Apps
None installed. Browse Grafana.com
Installed Panels
None installed. Browse Grafana.com
Installed Datasources
None installed. Browse Grafana.com

4. Analyse the available metrics


K8s / USE Method / Cluster
Last 1 hour Refresh every 10s
datasource
prometheus
CPU
CPU Utilisation
CPU Saturation (Load1)
Memory
Memory Utilisation
Memory Saturation (Swap I/O)
cloudv199.zdv.uni-mainz.de
cloudv200.zdv.uni-mainz.de
cloudv201.zdv.uni-mainz.de
cloudv202.zdv.uni-mainz.de
cloudv203.zdv.uni-mainz.de
Disk

For viewing alerts:

1. Connect to the Prometheus Alert Manager at https://alertmanager-main-openshift-monitoring.prod.openrisknet.org/#/alerts
2. Login as one of the OpenRiskNet administrators
3. View the current alerts. This screenshot shows only one alert firing, the "Dead man's switch" which should always be firing and can be used to check that the alerting system is active



alertname="DeadMansSwitch"

11:28:00, 2019-11-07

severity="none" prometheus="openshift-monitoring/k8s" cluster="prod.openrisknet.org"

4. As mentioned previously, alerts are also sent to the project's Slack channel so that admins are quickly alerted of any issues. Other mechanisms such as email can also be configured.

Contributions to e-infrastructure landscape

A lot of the contributions from WP2 are not only valuable within the OpenRiskNet project but are also contributions to the general advancements of e-infrastructures in Europe and the rest of the world.

- All software tools developed and code produced within the project is openly available from Github, meaning it will be available for the future and can be directly used in other settings.
- The tools in OpenRiskNet are based around the use of containers and targeted at deployment though Kubernetes, which has become a standard approach over the life of the project, meaning they will be readily deployable to future environments.
- Many of the design documents produced and communicated can serve as blueprints and good examples for other projects adopting service-oriented and especially microservice architectures.
- The semantic annotation of APIs is general can easily be extended into other domains, and the implementations for discoverable APIs together with the registry are important and domain-agnostic solutions provided to the community.
- The cloud agnostic virtual environments can be directly reused in other domains and settings, there is by design a clear border between general solutions and OpenRiskNet-specific developments.
- The developed OKD-Orchestrator greatly simplifies robust deployment of

OpenShift and VEs on different types of infrastructures.

Issues and Challenges

Creating and maintaining the infrastructure encountered a number of challenges. Formost in this was fragility in the underlying OpenStack infrastructure on the SSC. This included:

- Large scale server shutdown because of loss of cooling
- Large scale server shutdown because of a lightning strike
- Sporadic networking problems
- Failure of VMs

We worked intensively with the SSC staff to fix these problems as best we could, but they resulted in some downtime and a greater than expected amount of maintenance effort.

IM has also seen similar fragility on other OpenStack environments which they use in other projects. Even if this shows that this problem is not unique to SSC, the decision was made to move to another, easily sustainable hosting environment controlled by one OpenRiskNet partner. Up to the date of this writing (November 2019), the infrastructure at JGU has proven to be robust for the 2 months it has been in use.

Additionally there was fragility in the OpenShift software stack, which compounded the problems. Key issues included:

- Problems with provisioning caused by failure of installing the Software Defined Network (SDN). This was reported to the OpenShift team but never fully resolved. The result was that provisioning sometimes failed and had to be repeated.
- Problems with GlusterFS storage. GlusterFS appeared to be the ideal storage solution, but the version that shipped with OpenShift 3.7 had a number of bugs and other issues (recognised by Red Hat) which were partly resolved during 2019 by upgrading to version 3.11. The result was that a considerable amount of time was needed to maintain the storage system, though the redundancy provided by GlusterFS meant that there was never any data loss.
- Problems with certificate renewal. We use Let's Encrypt certificates as they are widely used and free. However they have a lifetime of 3 months so need to be renewed frequently. In principle, this renewal process for the API server certificates is straight forward but in practice each time certificates were renewed it caused some failures in the GlusterFS storage and other OpenShift components that needed manual resolution.

A log of the work performed maintaining the reference site was created and can be found here[1].

Despite these issues, OpenShift proved to be a robust environment in most aspects and we fully believe that Kubernetes is the platform of choice for a flexible, highly available infrastructure. Adoption of containerization and Kubernetes in almost all relevant infrastructures, including the European Open Science Cloud (EOSC) confirms the decisions on the technology made at the beginning and throughout the project and will facilitate the harmonization and integration with these larger initiatives.

In future projects, purchasing a support contract (e.g. for OpenShift through Red Hat)

[1] https://docs.google.com/document/d/1Mk4_z9tOWRw5q8M_kMsFef7ObdEsxQEfsKi6ryxZCPQ/edit

should be considered, as that should make the management of the infrastructure easier. Alternatively, the research infrastructures could be deployed on basic infrastructures as provided, e.g. by EOSC, which was not an option since these did not exist when OpenRiskNet started. We are in contact with the EOSC-hub service providers how such a solution could be established through their early adopter programme.

Another consideration is the complexity of the entire stack, from the physical hardware, through to the OpenStack Virtual Machines, the OpenShift/Kubernetes container orchestration layer and the partner applications. Knowledge is needed to handle this, and project members not having a strong Ops or DevOps background needed assistance in deploying their applications. To do this we:

- Created extensive documentation and examples in the GitHub repository and Wiki
- Created the "OKD Orchestrator" to significantly simplify the definition and formation of OpenSHift v3 clusters
- Organised several internal knowledge dissemination clinics
- Organised public webinars
- Organised an 'application deployment' workshop at the final GA meeting

We consider these efforts as 'ongoing work', and the knowledge and material generated in this project should be of great benefit to future projects. Additionally, this might also result in opportunities for SMEs to offer the deployment and hosting of OpenRiskNet virtual environments as a consulting service contributing to the sustainability efforts as outlined in more detail in the Dissemination and Exploitation plan.

RISKS AND MITIGATIONS

From the OpenRiskNet DoA and D2.3, the Risks that are deemed relevant to the current deliverable are R5, R6, R10, R11, and R12. In the table below, we comment upon these.

Table 2. Description of risk and proposed risk mitigation measures

Description of risk (level of likelihood: Low/Medium/High)	**Proposed risk mitigation measures**
R5: Technical advantages in computer hardware and software concepts will render the proposed concepts (microservices and containerisation) obsolete (medium to high)	We will constantly monitor the state-of-the-art of available deployment and virtualisation solutions and select the most suitable ones. If necessary, the infrastructure will be adapted to the changing standards. Update at M18: Microservices and Containerisation continue to be highly important components in modern e-infrastructures. If anything, their importance has increased during M1-18. Update at M36: This risk did not surface.
R6: Virtualisation options will not work with future hardware and software concepts (low)	Even if the underlying technology might change, the concepts of microservices and containerised applications are expected to be valid for the foreseeable future. Specific tools like DOCKER and MANTL can then easily be substituted with newer approaches, when these become available. Update at M18: Docker is still the most widely used containerisation implementation, but MANTL has been discontinued and OpenRiskNet has changed to use Kubernetes/OpenShift that has the highest momentum and is backed by Google and RedHat. Update at M36: This risk did not surface.
R10: Reference instance unstable due to national cloud providers not production-grade	We will work together with national cloud providers to pinpoint problems. We will also develop contextualisation protocols to overcome potential infrastructure stability gaps. We will also make the entire deployment process portable to allow for moving between cloud providers where we have sufficient resources. Update at M36: The national cloud provider SSC (Sweden) has caused problems to the project due to issues with stability. Because of our

	efforts in portability, the transfer of the reference implementation from SSC to JGU was smooth.
R11: Momentum in community shifts from OpenShift towards Kubernetes	OpenShift adds a layer on top of Kubernetes, we e.g. use the CI/CD in OpenShift and the KeyCloak service provided by RedHat. There is no conflict between OpenShift and Kubernetes, and in case OpenShift is discontinued we can shift towards Kubernetes. Update at M36: While Kubernetes has more momentum and a larger community than OpenShift, they are both active projects with large community support.
R12: The OpenRiskNet software stack becomes complex, the level of technical expertise needed is high, having an impact on sustainability.	Kubernetes is becoming more and more mainstream, and the pool of people using it continues to grow. This means that more information is made available online, and more examples and experienced people are available. Further, many tools and frameworks that simplify the ecosystem is emerging. We will stay updated on the recent developments in the field, educate the partners in the consortium, and document our infrastructure and setup for easier maintenance and sustainability. Update at M36: The stack is somewhat complex, but the documentation, automation and tools made available, such as the OKD Orchestrator, simplifies a lot and reduces the impact on sustainability.

CONCLUSION

This document presents the Reference OpenRiskNet system together with the updates in WP2 during RP2. The system is deployed at the Johannes Gutenberg Universität Mainz and is available at https://prod.openrisknet.org/. Users can log into this reference virtual environment (VE) using LinkedIn, GitHub or Elixir credentials removing the need of creating a new account. This gives access to the infrastructure and the deployed OpenRiskNet services. Specific accounts and access rights are needed only for administrative tasks.

Sustainability of the reference VE is guaranteed for at least 2 years and negotiations with other cloud infrastructure providers are ongoing to extend this time of operations (BEAR cloud solution at the University of Birmingham). Additionally, the availability of the sources, documentation and tutorials, as well as containers for the basic infrastructure and OpenRiskNet applications running on them is secured based on standard solutions like GitHub and OpenAire and in an increasing extent the EOSC-hub and marketplace.

GLOSSARY

The list of terms or abbreviations with the definitions, used in the context of OpenRiskNet project and the e-infrastructure development is available:

https://github.com/OpenRiskNet/home/wiki/Glossary

REFERENCES

1. Dudgeon T, Spjuth O, Bois F, Bachler D. Development infrastructure online (Deliverable D2.1). 2018. doi:10.5281/zenodo.1479139

2. Rautenberg M, Karwath A, Kramer S, Dudgeon T, Spjuth O, Bachler D, et al. Initial API version provided to providers of services (Deliverable 2.2). 2018. doi:10.5281/zenodo.1479444

3. Bachler D, Dokler J, Dudgeon T, Willighagen E, Karatzas P, Lynch I, et al. Final API available for internal and external service providers (Deliverable 2.4). 2019. doi:10.5281/zenodo.2597061

4. Spjuth O, Dudgeon T, Bachler D, Gebele D, Rautenberg M, Alvarsson J, et al. Report on deployment of virtual infrastructures with service discovery and container orchestration (Deliverable 2.3). 2018. doi:10.5281/zenodo.1479475

5. Floden E, Lloret-Villas A, Di Tommaso P, Spjuth O, Farcal L, Dudgeon T, et al. Compute and data federation (Deliverable 2.5). 2019. doi:10.5281/zenodo.3256306