OpenRiskNet

RISK ASSESSMENT E-INFRASTRUCTURE

Deliverable Report D2.3

Report on deployment of virtual infrastructures with service discovery and container orchestration

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	Douglas Connect GmbH
Star date	1 December 2016
End date	30 November 2019
Duration	36 Months
Project Partners	P1 Douglas Connect GmbH Switzerland (DC) P2 Johannes Gutenberg-Universitat 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) 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 2.3 Report on deployment of virtual infrastructures with service discovery and container orchestration
Deliverable Type	Demonstrator
Dissemination Level	Public (PU)
Work Package	WP2
Task(s)	Task 2.3, 2.4, 2.5, and 2.6
Deliverable lead partner	UU
Author(s)	Ola Spjuth (UU), Tim Dudgeon (IM), Daniel Bachler (DC), Denis Gebele (JGU), Micha Rautenberg (JGU), Jonathan Alvarsson (UU), Pantelis Karatzas (NTUA), Egon Willighagen (UM), Chris Evelo (UM), Marvin Martens (UM), Thomas Exner (DC) Reviewed by Barry Hardy (DC)
Status	Final
Version	V1.0
Document history	2018-05-15 Draft version 2018-05-31 Final version

Table of Contents

Summary

This report documents the Demonstrator for the Deliverable 2.3, describing the deployment of virtual infrastructure and applications making up the OpenRiskNet Virtual Research Environment (VRE). It outlines the system analysis, deployment fundamentals, service discovery, and a list of the currently available services. The production reference instance is deployed on the Swedish Science Cloud (SSC), and end user access is available at https://home.prod.openrisknet.org.

This deliverable includes work performed towards the following tasks in the DoA:

- Task 2.3: Establish security environment
- Task 2.4: Services discovery
- Task 2.5: Deployment of virtual infrastructures and container orchestration frameworks
- Task 2.6: Establishment and maintenance of OpenRiskNet reference instance

Introduction

The OpenRiskNet Consortium develops the OpenRiskNet e-infrastructure for the harmonisation and improved interoperability of data and software tools in the area of predictive toxicology and risk assessment. It will combine interoperable web services providing data or analysis, processing and modelling tools communicating over well-defined and harmonised application programming interfaces (APIs). The services will be deployable within an OpenRiskNet Virtual Research Environment (VRE)[1,2], and OpenRiskNet will provide resources to enable users to instantiate their own virtual infrastructures populated with the applications and middleware making up the VRE on public or private cloud resources, as well as in-house server/workstations. The OpenRiskNet e-infrastructure and VRE will support many aspects of risk assessment by allowing for the integration of all toxicology-related data sources, for the implementation and execution of processing and analysis pipelines and for the execution of modelling workflows.

System analysis and description

The OpenRiskNet infrastructure builds fundamentally on containers[3,4] as a core technology for enabling portable, isolated environments for applications that can be exposed as services within the e-infrastructure. In order to manage multiple containers on distributed resources (compute nodes), a container orchestration system is needed to accommodate for starting, stopping, scheduling, and packing containers on a set of compute nodes. The most widely used container orchestration systems include Docker Compose, Docker Swarm, Kubernetes, and OpenShift. The OpenRiskNet project considered these and other options (*see Deliverable 2.1*) and landed at a decision that at present the most suited for our needs are either Kubernetes (backed by Google) or OpenShift (Red Hat's distribution of Kubernetes with important extra capabilities). While we kept the door open towards Kubernetes, the current development and production environment is based on OpenShift.

While containers and container orchestration can be run on single computers such as servers, virtual infrastructures (comprising compute nodes, networks, storage, etc.) in elastic environments such as available cloud computing providers, offer compelling properties regarding cost effectiveness and improved resilience. A fundamental design decision in OpenRiskNet is to implement the e-infrastructure as a Virtual Research Environment (VRE), consisting of the deployment of virtual infrastructure, the middleware in terms of container orchestration software and workflow engines, as well as a suite of compatible tools packaged as docker containers with the APIs defined to sustain semantic interoperability between the services.

The OpenRiskNet consortium is providing its own reference VRE that acts as the public showcase for the project, but users and organisations should also be able to launch OpenRiskNet VREs on their own resources. In order to achieve this, a high grade of automation on all levels of the infrastructure is needed. With such complexity comes the importance of testing, and continuous integration (CI) and continuous deployment (CD) has been set up by the OpenRiskNet consortium to facilitate continuous packaging, testing, and deployment. The development infrastructure and CI/CD system has previously been reported in D2.1.

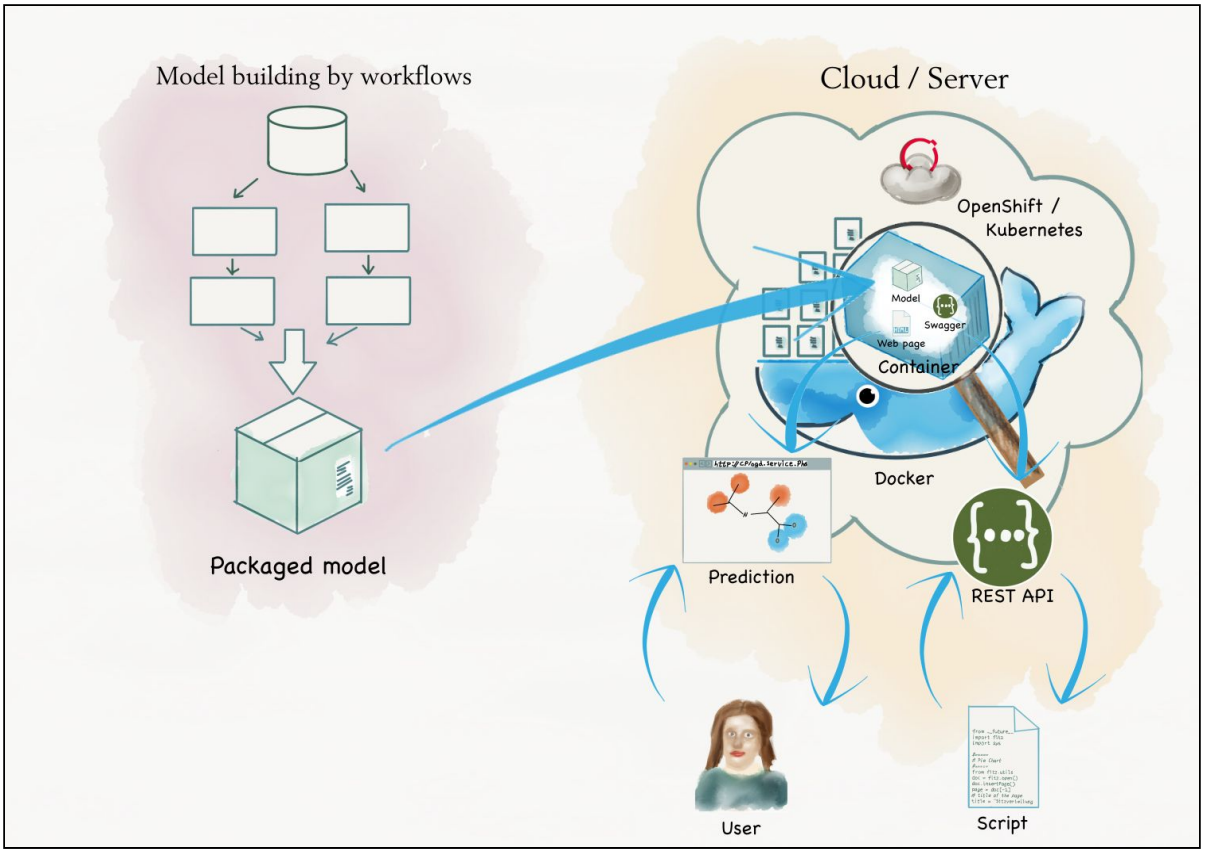

Figure 1: *An example case of the use of OpenRiskNet VRE for predictive models. Scientific workflows can be used to automate model building components, with the resulting model packaged as a container. The container can then be deployed in the OpenRiskNet VRE as a service/pod in OpenShift. The service can then provide end-user services, e.g. in the form of web-based interfaces (that can be consumed by humans) and REST APIs (that can be consumed via scripts).*

Demonstrator Access

This is a 'demonstrator' report. As such we concentrate primarily on describing how the output of the project can be demonstrated at the Month 18 stage. The report was mostly written in May 2018 as the project approached its mid-point. At this stage the project is in its most active stage, and if the report is read even a few weeks later, some aspects might have changed significantly. To accommodate this, we try to provide basic information in this report, including screenshots, but to link out to live sources of information such as GitHub repositories and Google Drive which will continue to be updated.

In order to demonstrate many of the systems in the production site a login will be needed. There are three types of user accounts.

1. OpenRiskNet partner applications that have been deployed to the site can be accessed as an end-user. This is straightforward as individuals can login using one of the social authentication providers such as GitHub or LinkedIn. Information on how to do this is available from the reference site home page: https://home.prod.openrisknet.org.
2. Access to the OpenShift console, API and command line tools requires developer access. This is controlled through the OpenRiskNet GitHub organisation and a small number of dedicated admin accounts. Viewing these parts is best done under guidance of a developer experienced with the system.
3. For the demonstrator reference instance, access to administrative areas of the cloud provider and OpenShift will not be possible as access needs to be tightly controlled. Instead we will arrange for a guided tour of these areas on request.

The documentation for the project is primarily in GitHub projects owned by the OpenRiskNet organisation (https://github.com/OpenRiskNet). All these documents are publicly accessible under open licenses.

Deployment

Virtual Infrastructure and OpenShift cluster

The OpenRiskNet VRE requires a virtual infrastructure to be instantiated on either a public cloud (IaaS) provider or a local computer resource. The procedures for creating the OpenShift cluster suitable for OpenRiskNet VRE on a virtual environment such as the SSC OpenStack environment (and in future other types of sites such as Amazon Web Services (AWS)) are described here:

https://github.com/OpenRiskNet/home/tree/master/openshift/environments

The procedures for deploying the individual OpenRiskNet infrastructure components and partner applications are described here:
https://github.com/OpenRiskNet/home/tree/master/openshift/deployments

The purpose of this deployment documentation is to provide descriptions of how to deploy these applications to your own Virtual Research Environment (VRE). The aim is to make this as simple and automated as possible by making these applications available from the OpenShift service catalog so that they can be deployed using the OpenShift web

console as well as using command line tools.

Automating deployment

In order to enable user-initiated instantiation of the OpenRiskNet VRE, we use KubeNow (https://github.com/kubenow/KubeNow), a cloud-agnostic platform for microservices, based on Docker and Kubernetes that was developed within the H2020-PheNoMeNal project (http://phenomenal-h2020.eu/home/)[5]. KubeNow provides a seamless mechanism to set up ready-to-use Kubernetes-based research environments, aimed to support on-demand scientific workloads. We extended KubeNow with functionality to instantiate OpenShift clusters, and set up the infrastructure applications needed for OpenRiskNet VRE. The present version is not feature-complete, with some functionality missing for different cloud providers, and the remaining features will be added during M19-M30.

Services and templates

A key aspect of automating the provisioning of applications is through OpenShift Templates. These define all aspects of an application, and typically consists of at least three components:

1. a DeploymentConfig that defines the Docker images to run and the options for running them in a Pod;
2. a Service that provides access to the running Pods from within the cluster; and
3. a Route that provides access to a Service from outside the cluster.

Templates often define additional components of the running application such as configuration and persistent storage.

The aim is that by defining an appropriate template the provisioning of a complex application can be turned into a ‘one click’ operation. OpenShift itself uses this extensively to provision parts of the OpenShift infrastructure, and some of these applications are quite complex.

In some cases, these templates are complex, and significant work was spent on providing information and training to project partners in getting up to speed with generating these. In particular we have:

1. Created extensive documentation in the OpenRiskNet Home GitHub repository that can be found here: https://github.com/OpenRiskNet/home/tree/master/openshift
2. Run a regular series of on-line clinics where partners have been able to share experiences, material for which is located in the project’s Google Drive: https://drive.google.com/drive/folders/1A98zl9P0387ifTsI9w2QP0linGX6lTwq

Our aim is to describe and establish best practices for deploying containerised applications to a VRE. Some examples are:

1. The ‘example-java-servlet’ project that contains a very simple ‘Hello World’ type project and aims to demonstrate best practice for how to build and deploy it to a VRE: https://github.com/OpenRiskNet/example-java-servlet
2. Deployment of the Squonk application (IM) to a VRE, which is an example of deploying a complex application involving multiple containers and incorporating many of these best practices: https://github.com/OpenRiskNet/home/tree/master/openshift/deployments/squonk

3. LAZAR predictive toxicology service benefits from the automatic deployment whenever the source code is changed. For this a trigger chain is used which creates a docker image including the changed code with all dependencies and pulls to re-deploy the service.
4. Jaqpot predictive and modeling services, also benefits from this deployment strategy. Jaqpot consists of many microservices that are dockerised and the manual deployment of these was very time consuming. With Openshift this became easier by simply creating a template that handles the deployment, with all the necessary components needed by the services. After deploying the template, the user can customise the deployment in many ways such as scaling up and down on premise. The Jaqpot deployment template can be found at:

   https://github.com/KinkyDesign/KubeAndOpenshiftTemplates

One of the most important aspects that OpenShift provides above and beyond Kubernetes is an integrated CI/CD infrastructure. This allows to continually update applications as they are developed. Docker images are rebuilt and published to repositories so that they can be automatically deployed in any VRE. OpenShift provides multiple ways to achieve this, and we are setting up partner applications to utilise this. Examples include:

1. Simple build and deployment configs that update the Landing page that lists the partner applications that are deployed whenever the website material is updated in GitHub. A commit to GitHub with new material results in the web server being redeployed with the new material within a few minutes without the need for any human intervention. See the Home deployment and build of the openrisknet-infra project https://prod.openrisknet.org/console/project/openrisknet-infra/overview
2. The Squonk CI/CD project which runs Jenkins to build the Docker images for Squonk, ultimately resulting in redeployment of the application when updates are committed to the GitHub repository: https://prod.openrisknet.org/console/project/squonk-cicd/overview.

The work on this is ongoing. Admin access is needed for both of these sites.

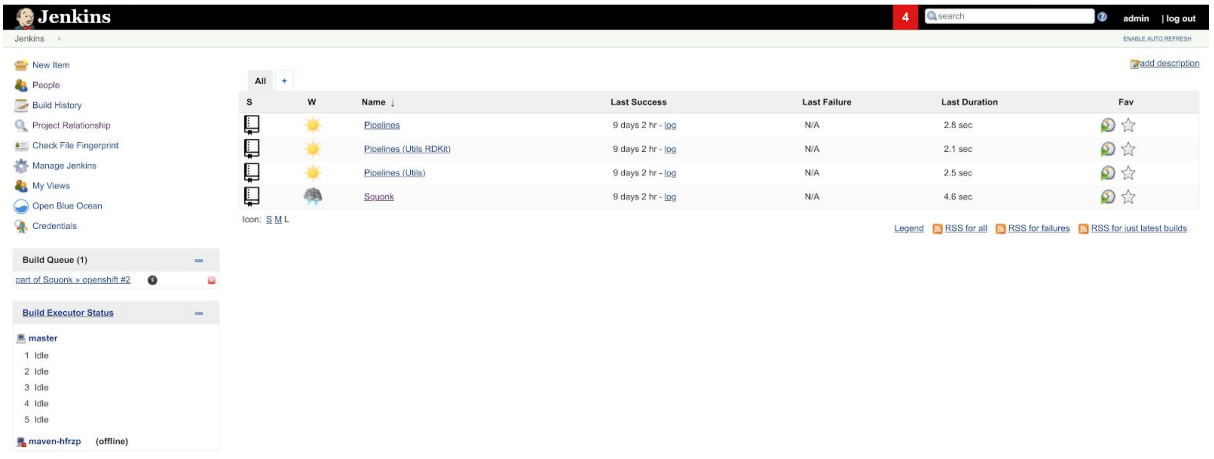

Figure 2: *Screenshot from the Jenkins continuous integration service as part of the CI/CD process for building the Squonk Computational Notebook and other projects used by Squonk. This will ultimately allow automated updating of applications as changes to their source are made.*

Security and encryption

Running web-based applications over HTTPS is nowadays the norm. This provides encryption of traffic in transit, and provides some guarantees over the authenticity of sites. To achieve this, TLS certificates have to be obtained from an authority that is trusted by the web browser. Historically this has been complex and expensive to achieve.

OpenRiskNet is taking advantage of two recent developments in this area that make the process of securing the OpenShift Routes (a Route provides external access to services running on the cluster) with HTTPS:

1. Let's Encrypt (https://letsencrypt.org/) allows to generate trusted certificates free of charge. Let's Encrypt is run by the non-profit Internet Security Research Group. Certificates have a 3-month lifetime making it highly desirable to automate the generation and renewal process.
2. ACME Controller (https://github.com/tnozicka/openshift-acme) is a project that automates the generation of TLS certificates for OpenShift routes.

By providing the ACME Controller as part of the infrastructure of a VRE, securing routes with TLS becomes remarkably simple. The route definition just needs to have the *kubernetes.io/tls-acme: "true"* annotation added to its definition and the ACME Controller will generate and deploy TLS certificates and renew them before they expire. We aim to have all public routes secured with TLS.

Traffic within the OpenShift cluster is also secured using TLS certificates. In this case, as this traffic is entirely internal to the cluster, it does not need to have a public chain of trust, and so self-signed certificates are used.

The OpenShift Console and REST API are publicly-accessible services. These are secured using a master certificate from Let's Encrypt. OpenShift provides a mechanism for updating these certificates.

Internal traffic between OpenRiskNet applications currently mostly runs over HTTP as it is private to the cluster. We have established a mechanism for securing this traffic with mutual TLS should that be required. Details can be found here: https://github.com/OpenRiskNet/home/blob/master/openshift/recipes/service-communication-via-tls.md

By default, data in an OpenRiskNet VRE (e.g. database files) is not encrypted on disk as it is private to the cluster, but most cloud providers, including OpenStack, provide a simple mechanism to encrypt volumes, so this would be easy to provide if necessary.

General infra applications and databases

As part of the process of deploying a VRE we include key infrastructure components that will be used by many of the applications that will be deployed. This avoids the developer from needing to provide and manage these components e.g. backup and restore will be handed centrally by the VRE administrators. Currently these infrastructure components include:

- PostgreSQL database
- RabbitMQ message queue
- Red Hat Single Sign On for authentication and authorisation

- ACME controller for generating and maintaining TLS certificates

Additional infrastructure components may be added during the project.

Instructions for deploying these components can be found here:
https://github.com/OpenRiskNet/home/tree/master/openshift/deployments/openrisknet-infra
https://github.com/OpenRiskNet/home/tree/master/openshift/deployments/acme-controller

Monitoring, logging and auditing

Consolidated logging and metrics can be enabled for a VRE by deploying some additional components. These provide a powerful way to know what is happening on the cluster and are very useful for troubleshooting. In a distructured multi-server environment this would otherwise be hard to achieve. This comprises the following components:

1. Consolidated logging using the ElasticSearch, Fluentd and Kibana (the EFK stack). This consolidates all the logs from the running hosts and containers into an ElasticSearch database that can be queried using Kibana (creating dashboards is also possible)
2. Consolidated metrics using Hawkular. This is OpenShift's internal mechanism for scraping metrics from pods to that CPU; memory and network usage can be monitored
3. Metrics using Prometheus, which is a more comprehensive metrics solution that can gather host and container metrics, and can provide automated alerts (e.g. about failures or disks becoming full). Dashboards can be generated with Grafana.

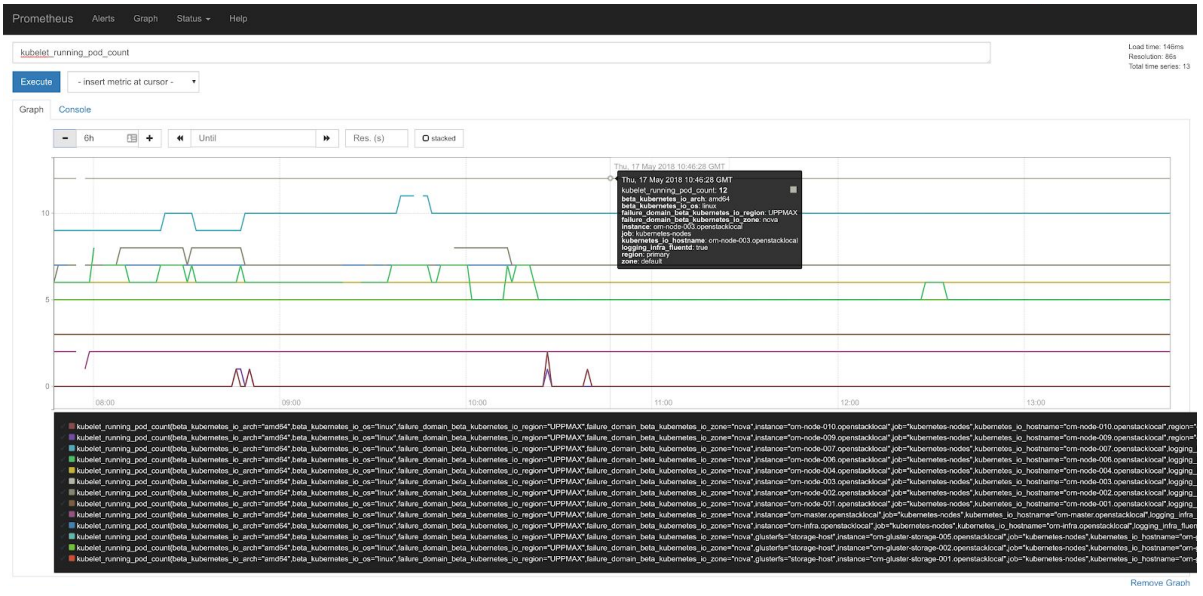

Figure 3: *Monitoring with Prometheus. This screenshot shows how visualisation of the number of Kubernetes pods running on each node in the OpenShift cluster is possible. A wide range of metrics can be monitored with Prometheus, and alerts can be generated from these metrics e.g. when a certain metric goes outside the expected normal range.*

Whilst consolidated logging, metrics and Prometheus have been deployed, we have yet to utilise these more. This will be done as the project progresses.

Security environment

The security environment is an integral part of an OpenRiskNet VRE. When a user creates a new VRE the security infrastructure is already in place, and applications that are deployed to the VRE can integrate into that security infrastructure.

However, this security environment needs to remain flexible, allowing the administrator to configure it according to their organisation's needs. An example would be the need to delegate authentication to the organisation's LDAP or Active Directory server.

We chose to use Red Hat Single Sign On (SSO) as the security environment. This is a Red Hat supported product, based on the upstream open source project Keycloak. This provides an enterprise grade security environment that is very flexible in how it can operate. We highlight two aspects:

1. Applications can be integrated into the security environment using OpenID Connect (an extension of OAuth) or SAML2. Both are widely adopted protocols supported by nearly all application servers. This allows OpenRiskNet partner applications to be easily integrated into the environment and benefit from the SSO environment. To date the Squonk (IM) and Jaqpot (NTUA) applications have been incorporated.
2. The ability to federate authentication to other sources such as LDAP or social providers such as Facebook, GitHub or LinkedIn. We have demonstrated this using the GitHub provider. This feature allows the administrator of a VRE to configure authentication according to their needs.

We have created documentation on how to set up the SSO environment as part of the OpenRiskNet VRE that can be found here:
https://github.com/OpenRiskNet/home/tree/master/openshift/deployments/openrisknet-infra

We ran a clinic for OpenRiskNet partners on 12-APR-2018 on the topic of how to integrate applications into the SSO environment.

Information on how to automatically incorporate the Squonk Computational Notebook into the SSO environment can be found here:
https://github.com/InformaticsMatters/squonk/tree/openshift/openshift/templates

Jaqpot REST services utilise the OpenID connect protocol that is offered by Keycloak. In order to do so, open source libraries certified by http://openid.net/connect/ foundation, have been used. This procedure and examples can be found in the clinic slides and can be found here:

https://docs.google.com/presentation/d/1-xxYFyrl5zaNyQsdQOfmQMqSoHalAy1LhmZsw9C9vHk

Administrative access to the SSO environment (https://sso.prod.openrisknet.org/auth/) is restricted to a small number of OpenRiskNet administrators.

Access to OpenRiskNet applications that are deployed to the VRE is handled through an SSO 'realm' (by default this realm is called 'OpenRiskNet'). Each application is registered to the realm as a 'client', and end users access those applications by logging into that realm using whatever mechanism is configured by the administrators, but usually requiring

them to enter a username and password. We have demonstrated that we can use GitHub as a social login provider, but this feature is not currently enabled. We are discussing how best to handle the end user authentication, with the preference to use a social login provider, possibly Linked In. Currently we use usernames and passwords handled by the SSO application and stored in the SSO database (users can potentially register themselves).

Another possibility is for Red Hat SSO to delegate to another OpenID Connect instance (e.g. another Red Hat SSO instance) allowing federation of users between VREs.

We have demonstrated enabling 2 factor authentication, with the user needing to generate a login token using the Google Authenticator mobile application. Currently we enable this only for administrative accounts.

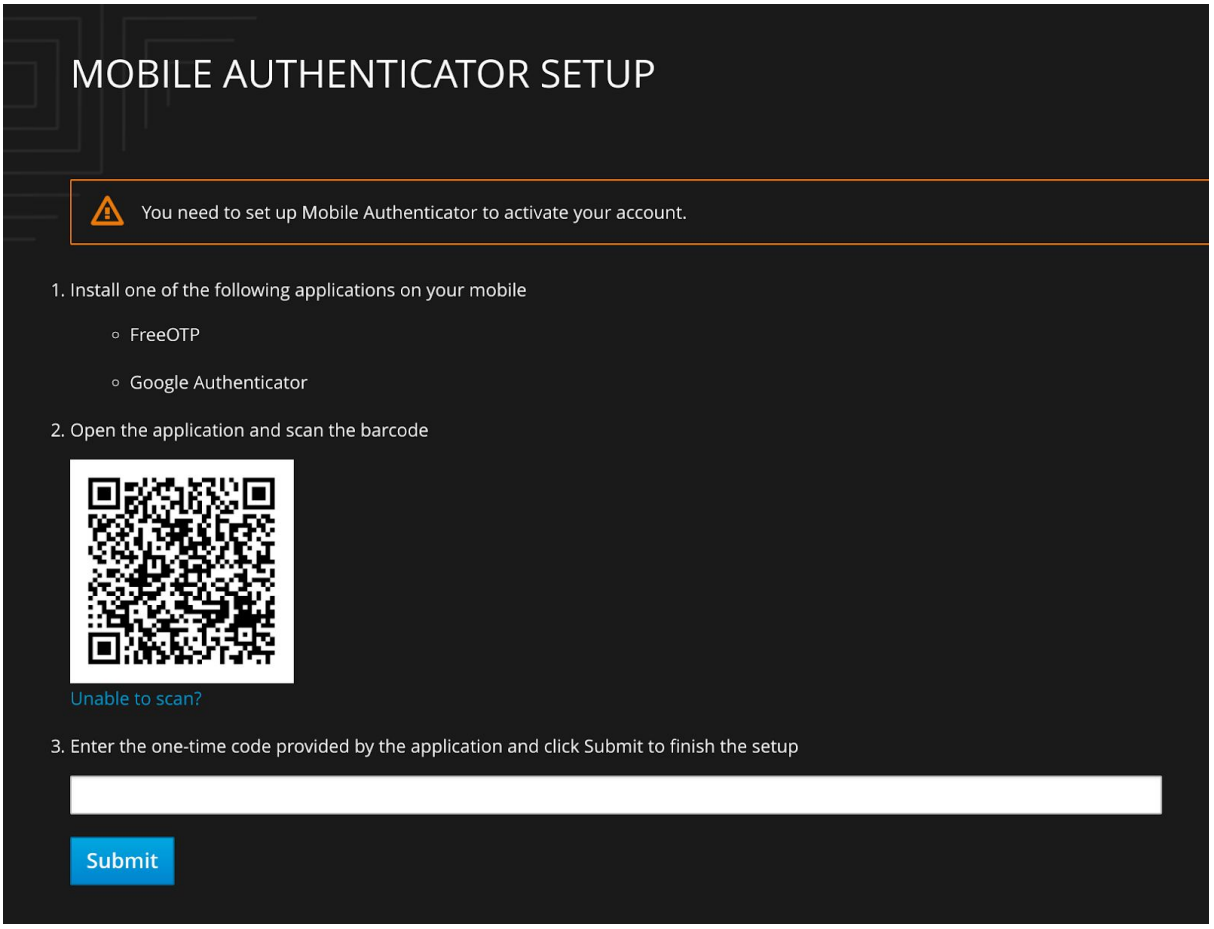

Figure 4: *Two factor authentication. For sensitive administrative accounts we add the requirement to enter an additional token that must be generated from a device such as a mobile phone. This provides significant additional security over just using a username and password.*

The SSO environment can be backed up by a number of means:

1. Export/Import of the 'realm' definition as a JSON file
2. Creation of a complete export during startup of the SSO application (this requires a small amount of downtime)
3. Backup of the PostgreSQL database that is used to store the information from the

SSO application.

Reference environment

In the D2.1 report provided at M6 we described the setup of a development environment that supported the basic software development activities of the project. This has now been superseded by a new production environment that provides the basis for the long-term reference site for the project, and the basis for creating a new VRE that could be created by project partners or third parties.

This production site is currently comprised as follows:

- Virtual server and networking environment provided through the OpenStack running on the Uppmax region of the Swedish Science Cloud. Currently this consists of:
    - 1 master node running the OpenShift API, controllers and etcd
    - 1 infrastructure node running the Docker registry and Router
    - 7 worker nodes dedicated to OpenRiskNet applications
    - 3 worker nodes dedicated to performing HPC-style work
    - 3 storage nodes providing persistent storage through GlusterFS
      (note: the worker nodes can be easily scaled up or down)
- OpenShift Origin cluster deployed to those servers providing a container orchestration platform and Continuous Integration and Continuous Delivery (CI/CD) capabilities
- Containerised consolidated logging and metrics management systems deployed to allow efficient monitoring and alerting of the cluster
- Containerised solutions for components that can be considered as OpenRiskNet 'infrastructure' components, including
    - Databases (PostgreSQL)
    - Message Queues (RabbitMQ)
    - Single Sign On (SSO) security infrastructure using Red Hat SSO (described in the previous section)
    - OpenShift ACME as a mechanism to secure external routes using TLS (HTTPS)
    - OpenRiskNet Service Registry for discovering OpenRiskNet services
- Partner applications including (see **Table 1** for more details):
    - Squonk Computational Notebook (IM)
    - Lazar predictive toxicology service (JGU)
    - JGU WEKA REST Service (JGU)
    - Jaqpot predictive modelling services (NTUA)
    - BridgeDB (UM)
    - CPLogD (UU)
    - Modeling Web (UU)

The OpenStack console is located at https://uppmax.cloud.snic.se/project/. An account at the SSC is needed to access this. The console provides access to the OpenStack environment and allows servers, networks and storage to be provisioned. This can also be performed using command line tools.

The following screenshot shows the servers for the production environment.

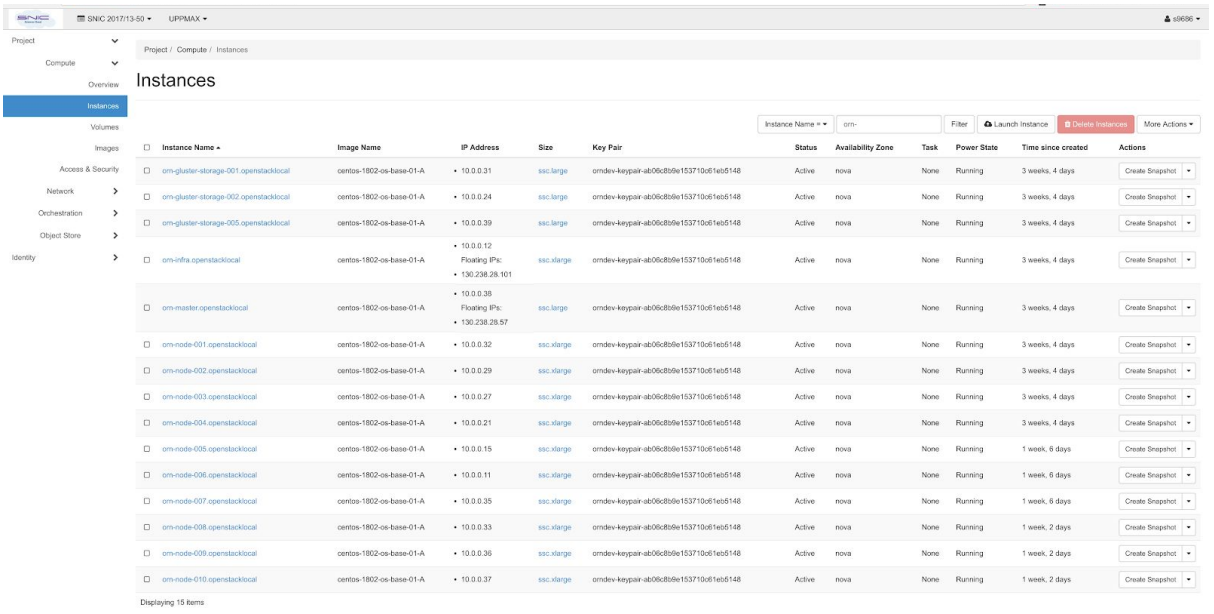

Figure 5: *A screenshot showing the servers for the production environment. The OpenStack environment on the SSC allows to provision virtual machines, networks and storage. This screenshot of the OpenStack web console shows the VMs that form the production site.*

The OpenShift web console can be accessed at this location: https://prod.openrisknet.org/. A developer account with the OpenRiskNet GitHub organisation is needed.

Once logged in, projects (Kubernetes namespaces) can be accessed based on user-specific access rights. For instance, the following screenshot is for the 'openrisknet-infra' project that contains containerised applications for key parts of the OpenRiskNet infrastructure, including the SSO application and the PostgreSQL database it uses.

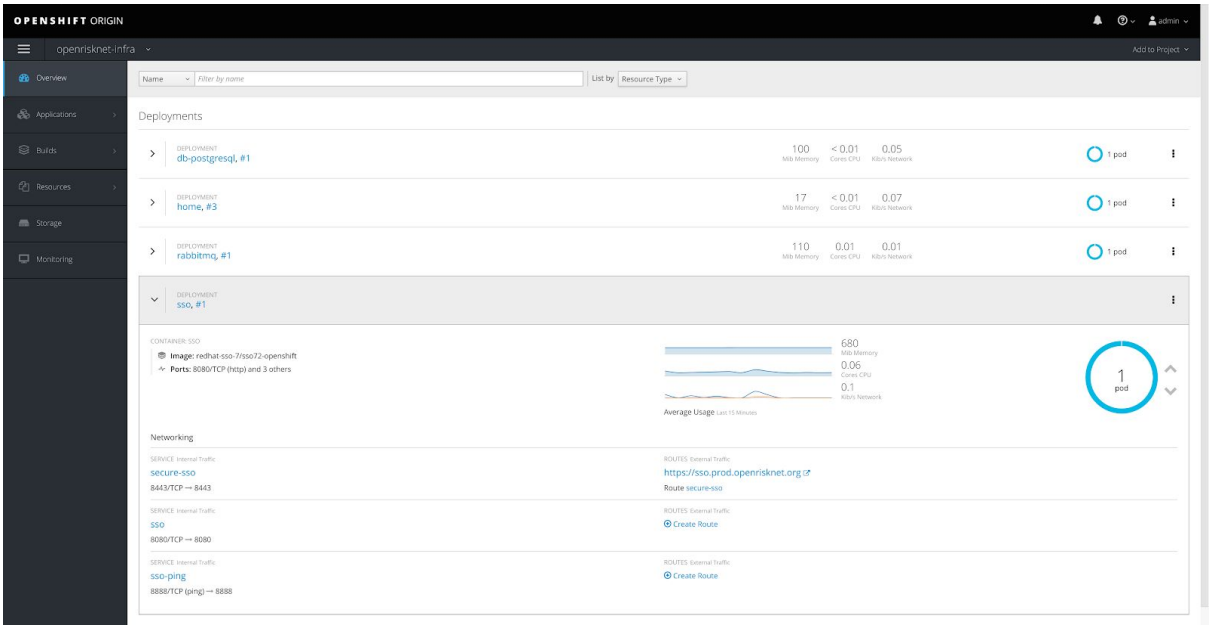

Figure 6: *OpenShift web console. This screenshot shows one of several projects from the production site. The console allows to inspect and update the applications running in these projects.*

A home page is available that provides links to the systems in the reference site that are described above: https://home.prod.openrisknet.org. Administrator rights will be needed to access some of these systems. Basic usage instructions can be accessed from that page.

A guide for deploying production-ready services to OpenRiskNet VRE can be found here: https://github.com/OpenRiskNet/home/blob/master/openshift/deployments/ProductionDeploymentGuide.md

The deployed applications and instructions for using them will be continually updated during the project and, in this way, managed externally to this report. Please refer to the home page for further information: https://home.prod.openrisknet.org

The end user support infrastructure is online (reported in Deliverable 3.2) and available on https://openrisknet.freshdesk.com/. Since the reference instance is only recently publicly available and we have made little public announcements on this, the support system is currently not in much use. However we anticipate that we will receive more requests and questions as the OpenRiskNet VRE is tested, and we are ready and prepared to handle these requests using the support system.

Service discovery

Aim

As outlined in the project proposal and then again in detail in the D2.2 report, the OpenRiskNet project anticipated the need for a service discovery component called the **OpenRiskNet Service Registry**. This component should run within every VRE and monitor the services in the VRE. When new OpenRiskNet services are started in the VRE these should be discovered by the Service Discovery component, their semantically annotated service description (called the **OpenRiskNet Service Description**) parsed and their semantic information indexed for future searches. Additionally, the cluster internal DNS name and port need to be stored in order for other services to be able to make requests against the other services' APIs.

OpenRiskNet Service Description

As outlined in the D2.2 report, an extensive search has been conducted to find suitable approaches for adding semantic information to the description of REST APIs. The major challenge in this case is that a REST API is a relatively complex interface that requires extensive specification to be used correctly (JSON schemata, MIME types, query parameters, header values, etc.) in addition to the needs of a semantic view on the inputs and outputs of the operations. After careful evaluation, a combination of two approaches was selected that extends the industry standard OpenAPI (formerly Swagger) and turns the main openapi.json description files into the semantic web standard JSON-LD by providing a JSON-LD context. This allows us to combine the tools of two existing, powerful, but so far independent sets of technology stacks: OpenAPI on the one hand to cover the technical details of a REST API with a big toolset around automatic creation of tools and documentation; and on the other hand JSON-LD and the semantic web technologies around the RDF data model and in particular SPARQL queries to query flexible knowledge graphs.

To facilitate experimentation and validate the feasibility of the OpenRiskNet Service Description concept, a web-based UI tool has been created called the OpenRiskNet Query Tester. It allows authoring of OpenRiskNet Service Description documents as well as the authoring and execution of SPARQL queries against these Service Descriptions: https://orn-query-test.cloud.douglasconnect.com/.

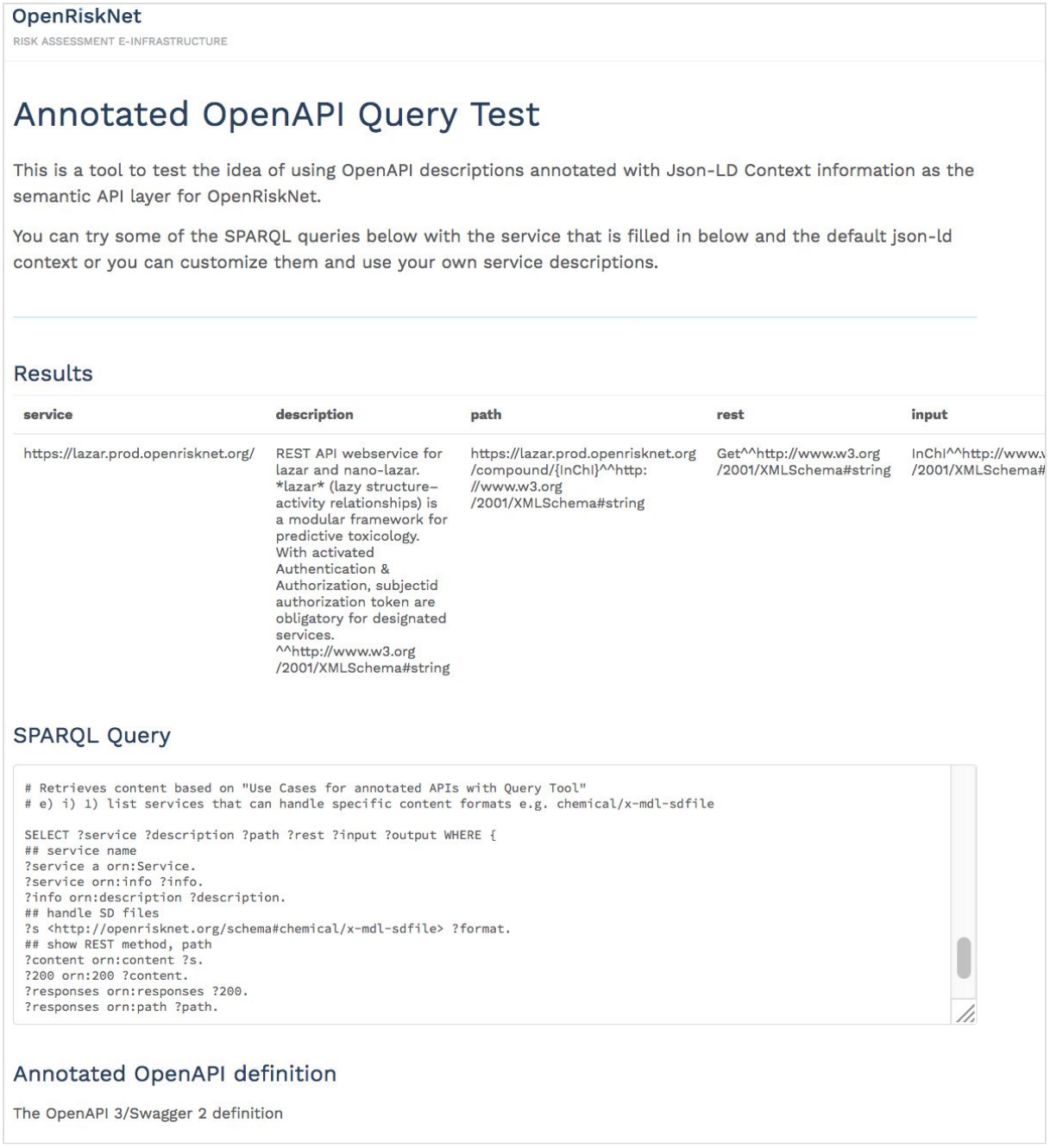

Figure 7: *Screenshot from the Annotated OpenAPI Query Tool that allows SPARQL queries against annotated APIs.*

For service providers, creation of an OpenRiskNet Service Description is a technically straightforward process if an OpenAPI/Swagger definition already exists for a REST API (which is a very common case). The creation of a document explaining the details of the semantic annotation for REST API authors is work in progress and will be available on the OpenRiskNet website.

After some experimentation, a minimal Json-LD context was created that allows a REST API to be treated as an OpenRiskNet-compliant service. This minimal context consists of a

set of alias definitions to make Json-LD keywords that are illegal Json keys for an OpenAPI.json/Swagger.json valid by prefixing with “x-orn” (lines 4 and 5 in the listing in Figure 8). To really use the power of semantic API annotation, every service provider should then additionally define a subset of the json-keys used in the response body and provide meaningful ontology terms for them (see the “smiles”, “inchi” etc examples in the listing below)

```
{"@context": {
    "@vocab": "http://openrisknet.org/schema#",
    "x-orn": "http://openrisknet.org/schema#",
    "x-orn-@id": "@id",
    "x-orn-@type":"@type",

    "smiles": "http://semanticscience.org/resource/CHEMINF_000018",
    "inchi": "http://semanticscience.org/resource/CHEMINF_000113",
    "inchikey": "http://semanticscience.org/resource/CHEMINF_000059",
    "cas": "http://semanticscience.org/resource/CHEMINF_000446"
  }
}
```

Figure 8: An example Json-LD context with four API specific keys defined as RDF predicates (in this case using terms from the CHEMINF ontology)

The Service Registry

A first iteration of the OpenRiskNet Service Registry was created and deployed into the reference environment (http://orn-registry-openrisknet-registry.prod.openrisknet.org/).

The Service Registry listens to Kubernetes events related to Kubernetes services to discover when new services become available inside the Kubernetes/OpenShift installation. The Kubernetes service description of a new service is then checked for the *“openrisknet-static-services”* annotation. If such an annotation exists it is expected to contain a link to the OpenRiskNet Service Description which is then downloaded, parsed and indexed. The service is then shown in the user interface of the service registry and SPARQL queries can be run against the information contained in the service description to find data based on the semantic annotations.

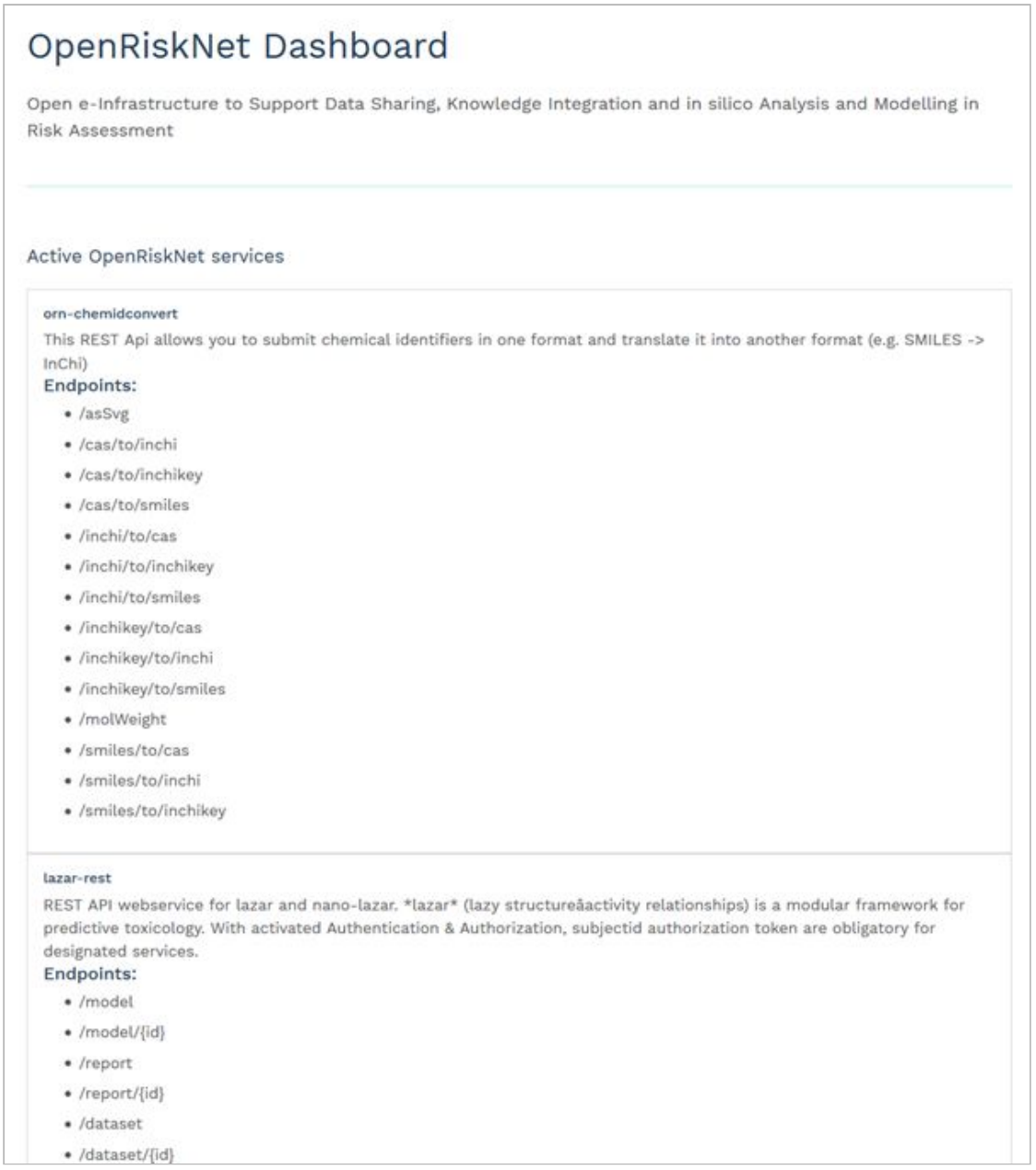

Figure 9: *OpenRiskNet Service Registry showing the lazar and chemidconvert services*

As of May 2018, the lazar modelling service (JGU) and the chemidconvert service (DC) have been adapted to serve a OpenRiskNet registry service compliant openapi definition. The other services are currently being modified to be findable via the Service Registry (see next section).

A document outlining the steps necessary to create a Registry Service compliant OpenAPI definitions is available in the documentation material of the registry:

https://docs.google.com/document/d/1a9Wndz5nqBzO2Km93lSpHjvftLLHufTo6Do3UpqyliE/

Available services

See below for a table and example screenshots (Figures 10-12) with details of partners' applications as services that so far have been deployed in the OpenRiskNet environment and featured in the Reference Instance. The services will be described in more detail in Deliverable 4.2 due at M24.

Table 1: *List of available services and links to partners' applications. The Resources column contains relevant links to resources for that application.*

Partner	Service	Resources
JGU	lazar https://lazar.prod.openrisknet.org/	Source code OpenShift template API definition Dockerfile Docker image
JGU	WEKA https://jguweka.prod.openrisknet.org/	Source code OpenShift template API definition Dockerfile Docker image
UU	cpLogD http://cplogd.prod.openrisknet.org/draw/	Source code OpenShift template: Uses s2i Dockerfile
UU	Modeling Web http://modelingweb.prod.openrisknet.org/	
IM	Squonk Computational Notebook https://squonk-notebook.prod.openrisknet.org/	Source code OpenShift template Docker image
NTUA	Jaqpot predictive modelling services https://api-jaqpot.prod.openrisknet.org/jaqpot/swagger/	Source code OpenShift template API definition Dockerfile Docker image
UM	BridgeDb https://bridgedb-swagger.prod.openrisknet.org/swagger/	Source code OpenShift template API definition Dockerfile Docker image

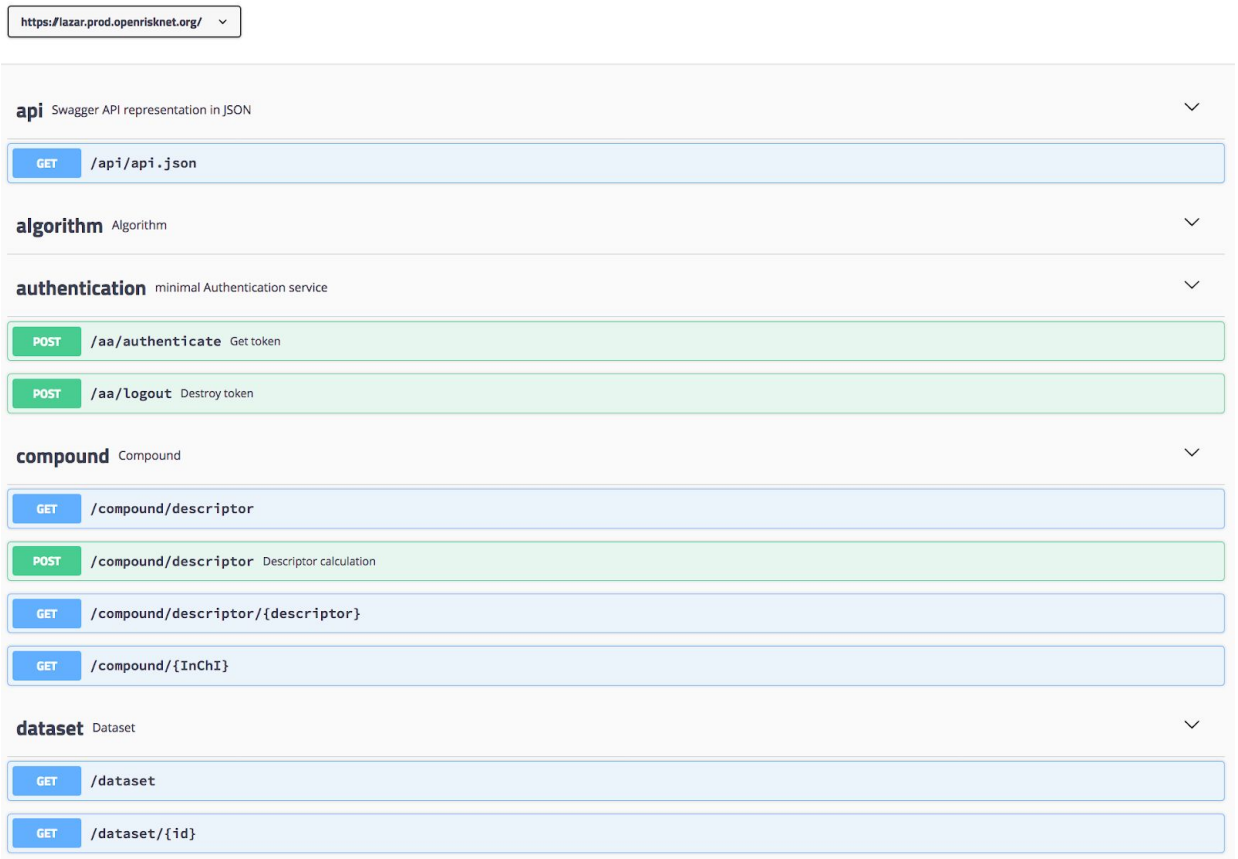

Figure 10: *Fully functional LAZAR predictive toxicology service implemented as a RESTful service with a SWAGGER UI as example application in the OpenRiskNet environment.*

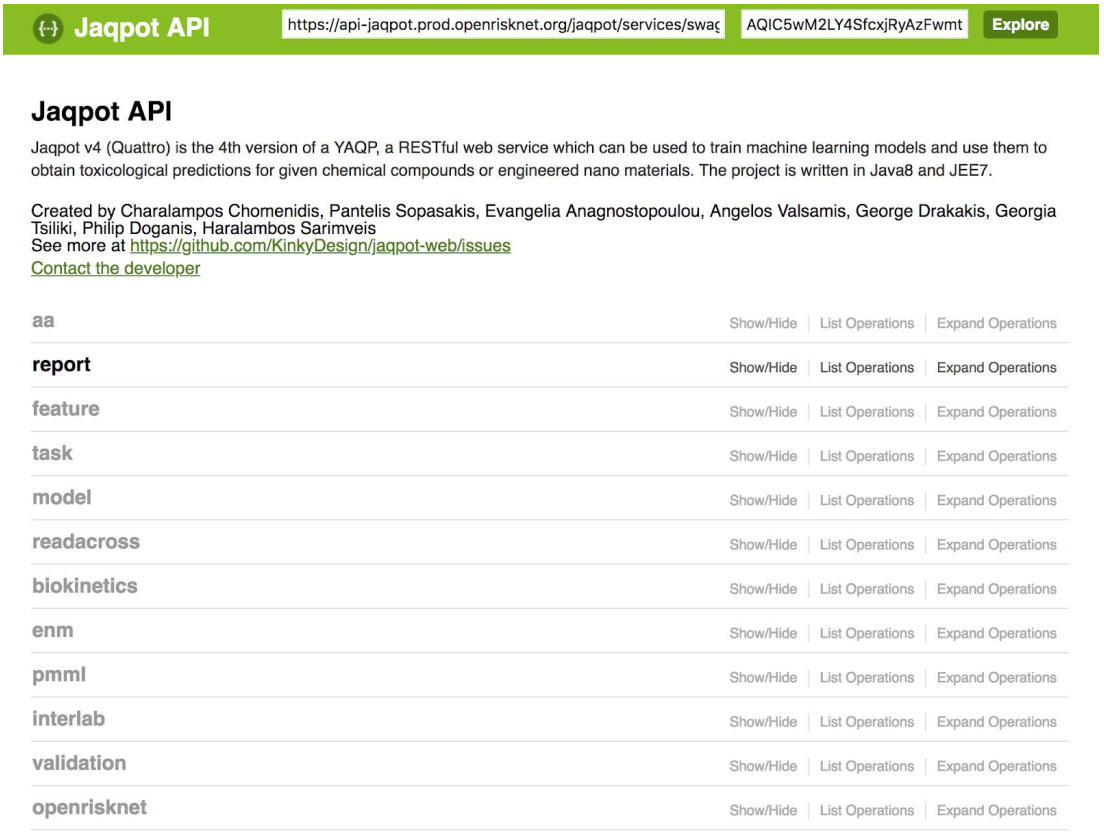

Figure 11: *Jaqpot REST services (Swagger documentation and Swagger UI) in the OpenRiskNet environment.*

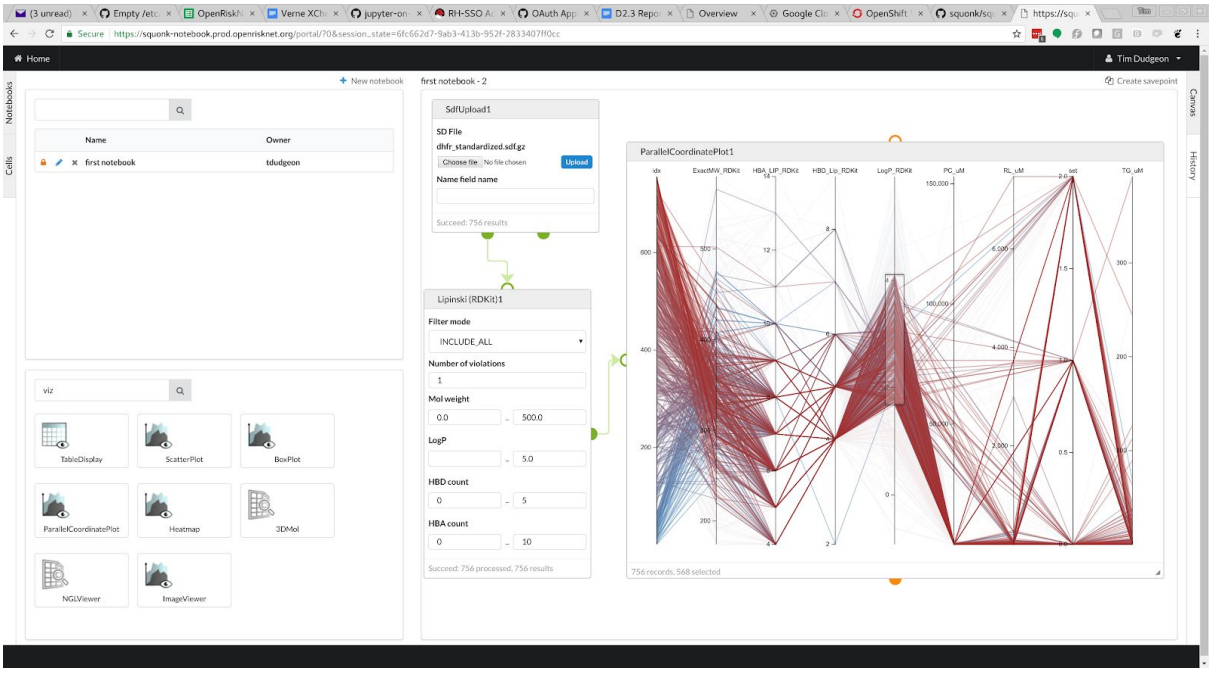

Figure 12: *The Squonk Computational Notebook running on the OpenRiskNet environment.*

Risks and mitigations

From the OpenRiskNet DoA, the Risks that are deemed relevant to the current deliverable are R5 and R6. In the table below, we comment upon these and also list additional risks that have been identified during M1-18.

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 stateoftheart 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.
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.
R10 (new): 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 entire deployment process portable to allow for moving between cloud providers where we have sufficient resources.
R11 (new): 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.

R12 (new): 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 more easy maintenance and sustainability.

Technical issues encountered and future work

The process of setting up the OpenShift cluster proved to be unexpectedly problematical due to fragilities in the underlying OpenStack cloud environment combined with fragilities in the OpenShift installation process. These problems are still being investigated and are yet to be fully resolved. These problems caused significant delay in generating this production cluster, and as a result a small number of aspects still remain to be completed:

- Provide and test backup and restore procedures;
- Provision high availability clusters;
- Automation of the orchestration process.

The delay affects the tasks 2.3, 2.4, and 2.5 that will need to be extended in time, but without any deviation in the total effort. The work will continue on these aspects until M24.

Conclusion

In this deliverable we describe the deployment process for OpenRiskNet virtual research environments, comprising a virtual infrastructure, an OpenShift cluster, OpenRiskNet infrastructure services, and operational container orchestration using the underlying Kubernetes cluster. We also describe the security environment, and the OpenRiskNet approach to service discovery.

The deliverable is in the form of a demonstrator manifested as the OpenRiskNet reference instance, deployed on SNIC Swedish Science Cloud (SSC) OpenStack resource, and available at https://home.prod.openrisknet.org/. The reference instance feature 10 services as of end of May 2018 and more to be integrated during the next months.

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

Abbreviation	Description
AAI	Authentication and Authorisation Infrastructure
CI	Continuous Integration
CD	Continuous Deployment
DoA	Description of Actions
IaaS	Infrastructure as a Service
SSC	Swedish National Infrastructure for Computing (SNIC) Science Cloud
SSO	Single Sign-On
TLS	Transport Layer Security
VRE	Virtual Research Environment

References

1. Candela, L., Castelli, D. & Pagano, P., (2013). Virtual Research Environments: An Overview and a Research Agenda. Data Science Journal. 12, pp.GRDI75–GRDI81. DOI: http://doi.org/10.2481/dsj.GRDI-013

2. Hurley DG, Budden DM, Crampin EJ. Virtual Reference Environments: a simple way to make research reproducible. Brief. Bioinformatics. 16(5), 901–903 (2015).

3. A. Silver, Software simplified, Nature, vol. 546, pp. 173–174, 05 2017.

4. Cito J, Gall HC. Using docker containers to improve reproducibility in software engineering research. In: Proceedings of the 38th International Conference on Software Engineering Companion - ICSE '16. ACM Press, New York, New York, USA, 906–907 (2016).

5. Payam Emami Khoonsari et al. Interoperable and scalable metabolomics data analysis with microservices. bioRxiv 213603; doi: https://doi.org/10.1101/213603