OpenRiskNet

RISK ASSESSMENT E-INFRASTRUCTURE

Deploying Applications to an OpenRiskNet Virtual Environment

The OpenRiskNet Consortium

OpenRiskNet: Open e-Infrastructure to Support Data Sharing, Knowledge Integration and *in silico* Analysis and Modelling in Risk Assessment
Project Number 731075

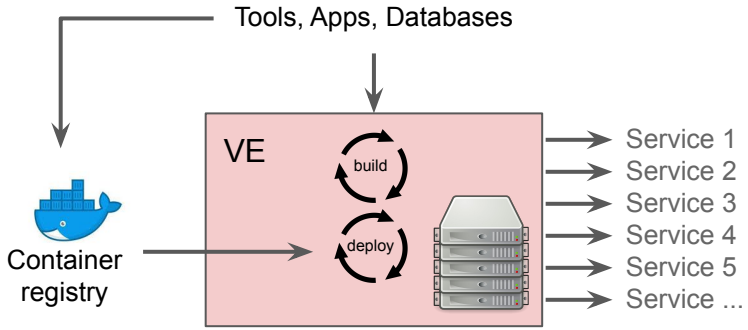
OpenRiskNet webinars series

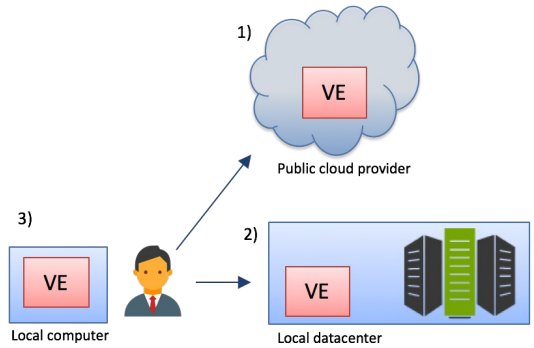
https://openrisknet.org/events/

	Topic	Date & Time
Past events	Introduction sessions to the OpenRiskNet e-infrastructure	Webinar recordings: • Session 1 (24 Sep 2018) • Session 2 (27 Sep 2018) • Session 3 (4 Oct 2018) • Session 4 (30 Oct 2018)
	Learn how to deploy the OpenRiskNet virtual research environment	Webinar recordings (25 Feb 2019)
	Demonstration on data curation and creation of pre-reasoned datasets in the OpenRiskNet framework	Webinar recordings (18 Mar 2019)
	Identification and linking of data related to AOPWiki (an OpenRiskNet case study)	Webinar recordings (26 March 2019)
	The Adverse Outcome Pathway Database (AOP-DB)	Webinar recordings (8 April 2019)
	How to describe OpenRiskNet services and their functionality by semantic annotation	Webinar recordings (13 May 2019)
	Use of Nextflow tool for toxicogenomics-based prediction and mechanism identification in OpenRiskNet e-infrastructure	Webinar recordings (27 May 2019)
	Demonstration on OpenRiskNet approach on modelling for prediction or read across (ModelRX case study)	Tuesday, **11 June 2019**, 16:00 CEST Registration: https://openrisknet.org/events/67/
	Combining neXtProt and WikiPathways strengths using SPARQL federated queries	Wednesday, **12 June 2019**, 20:00 CEST Registration: https://openrisknet.org/events/73/
Current event	Deploying Applications to an OpenRiskNet Virtual Environment	Monday, **24 June 2019**, 16:00 CEST Registration: https://openrisknet.org/events/66/
Future events	AOPlink workflow	Monday, **15 July 2019**, 16:00 CEST Registration: https://openrisknet.org/events/70/

OpenRiskNet Virtual Environment (VE)

- Computational infrastructure into which applications can be deployed
- Includes environment for building and testing those applications
- Includes compute, security, storage, monitoring ...
- Can be deployed to range of infrastructures



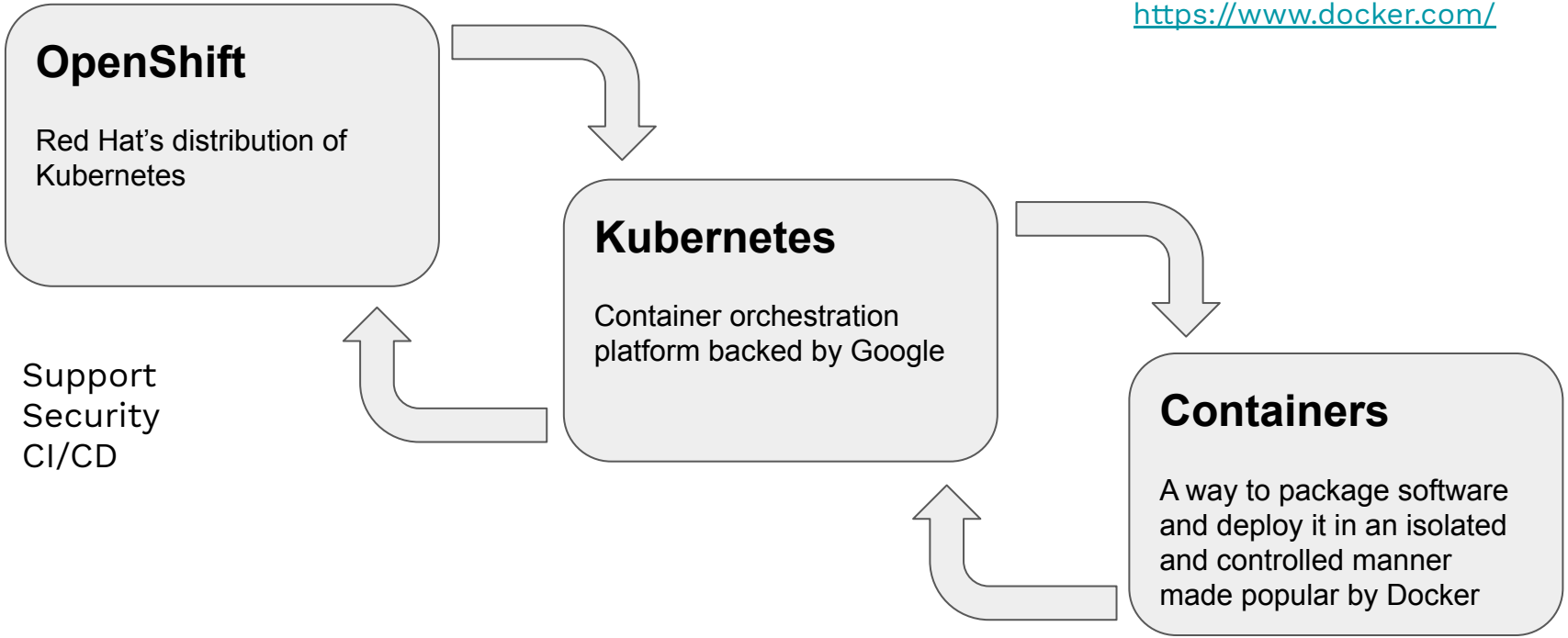


What forms a VE

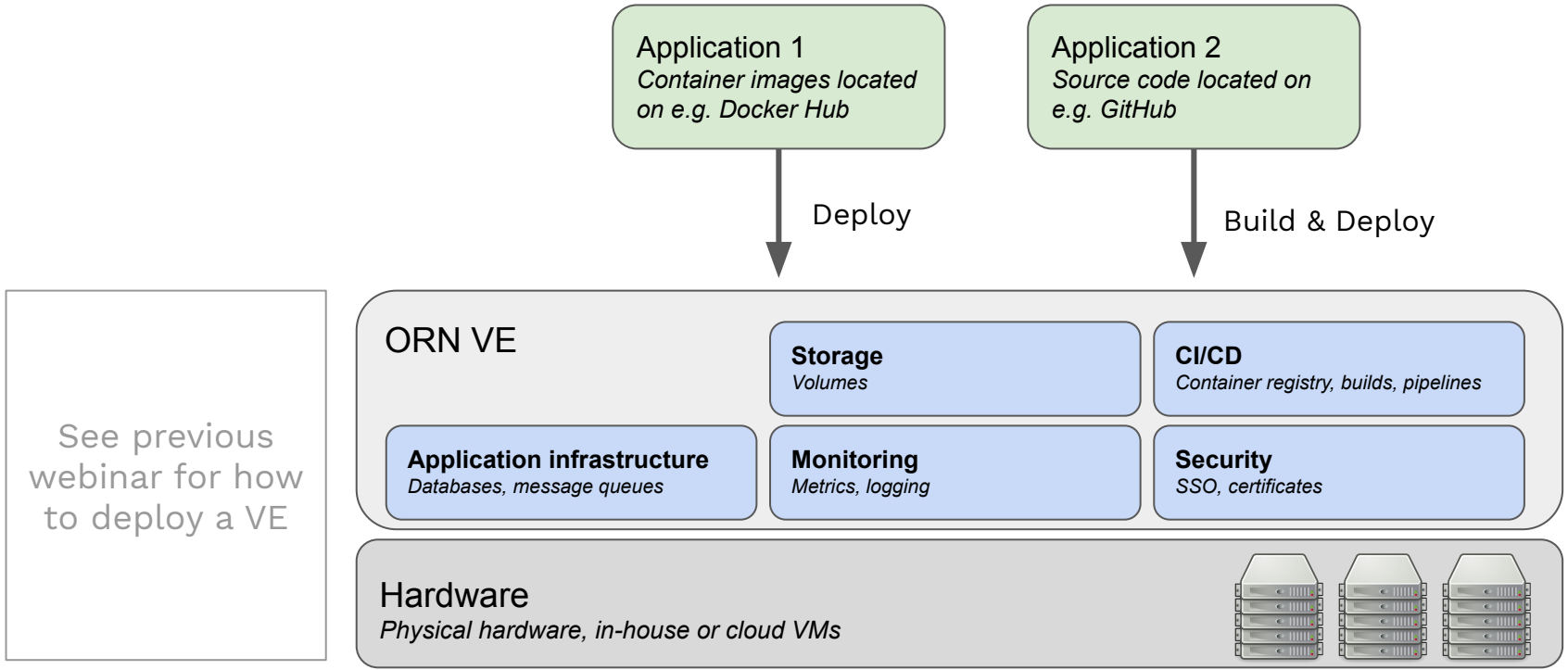
https://www.openshift.com/

https://kubernetes.io/

https://www.docker.com/



App Deployment to a VE



Introduction to containers

- A **container** is a set of Linux processes running in an isolated environment that is managed by features of the Linux kernel
- A bit like **virtual machines** but much more lightweight and efficient
- The software and data for those processes are packed into a container **image** that can be distributed

Packaging container images

- An image can package up pretty well anything you want
- Package multiple components into one container or each component into separate containers and let them communicate with each other
- Typically defined using a ‘Dockerfile’

https://github.com/alanbchristie/PySimple

```
FROM python:3.7.3-slim
WORKDIR /app
ADD . /app
EXPOSE 8080
RUN pip install -r requirements.txt
USER nobody
CMD ["python", "app.py"]
```

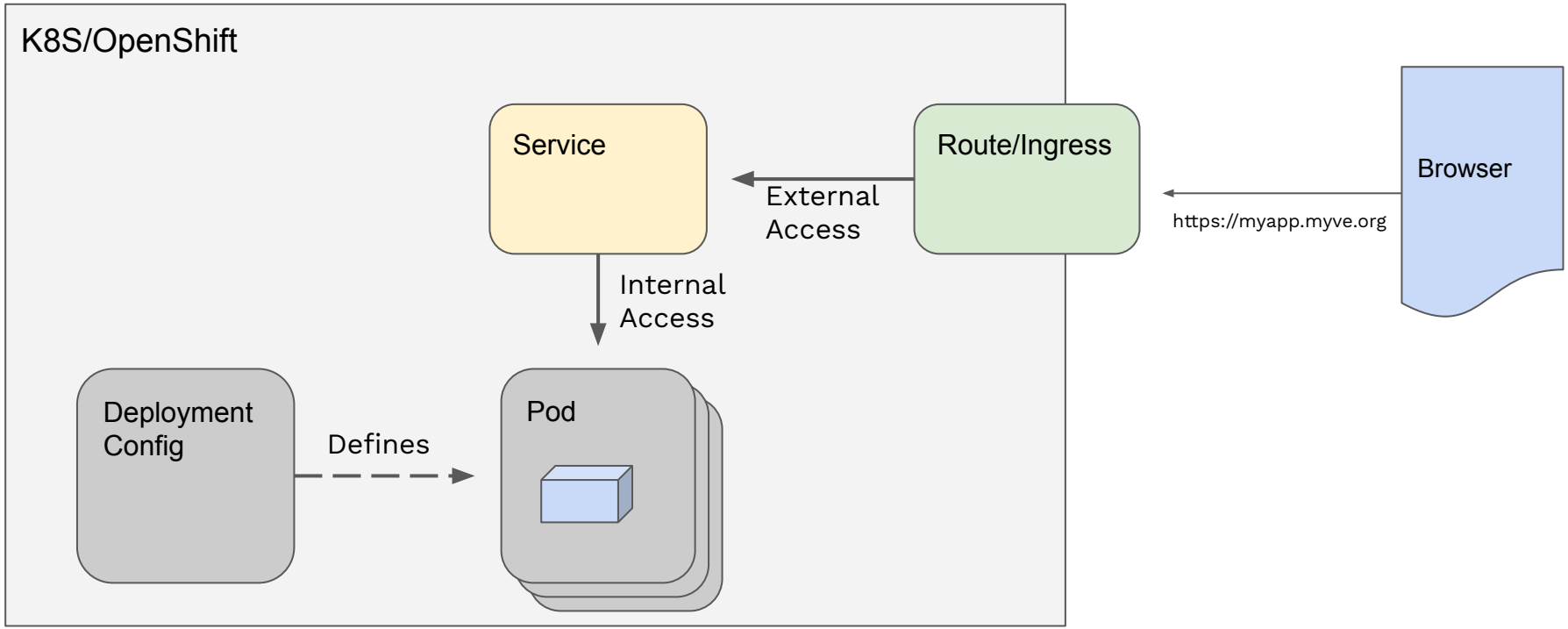
```
$ git clone git@github.com:alanbchristie/PySimple.git
$ cd PySimple
$ docker build -t tdudgeon/pysimple .
$ docker push tdudgeon/pysimple

…

$ docker pull tdudgeon/pysimple
$ docker run -d -p 8080:8080 tdudgeon/pysimple

$ curl http://localhost:8080/
```

Deploying container images on an ORN VE



Procedure for deploying applications

Step 1: Create your container images

Step 2: Deploy to OpenShift - multiple approaches possible

Step 1: creating container images

Create your container images externally and push to registry such as DockerHub

or

Use OpenShift's CI/CD mechanisms to build the container images and push to OpenShift's own container registry running in the VE

Step 2: Deploy to OpenShift

Multiple approaches possible

1. Web console vs. CLI vs. REST API
2. Manual/interactive procedure
3. Templates
4. Operators

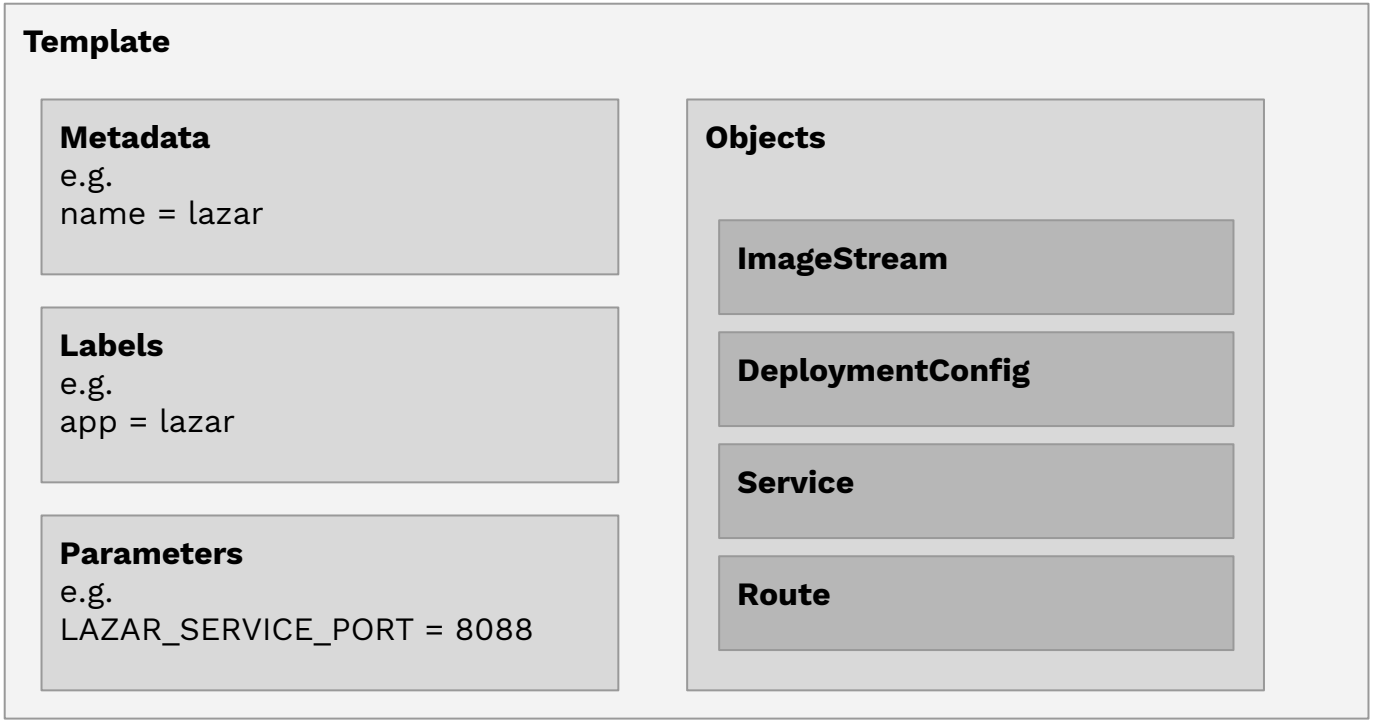
We will show some examples.

Deployment examples

1. Deploy app through web console
2. Deploy app using CLI
3. Deploy Lazar using CLI
   a. The Lazar template
   b. Deploying
4. Deploying Lazar from web console
5. Deploying Squonk using Ansible
   a. Templates
   b. Playbooks

Anatomy of the Lazar template

https://github.com/OpenRiskNet/home/tree/master/openshift/deployments/lazar



Lazar template - Parameters

```
parameters:
- name: IMAGE_TAG
  description:
    The lazar docker image tag
  value: latest
- name: ROUTE_NAME
  description:
    The name of the service endpoint.
    This is typically *lazar* but if you want different instances
    you can pass another name like *lazar-2*.
  value: lazar
- name: ROUTES_BASENAME
  description:
    The base name of the service endpoint.
    This is typically the PROD or DEV URL basename.
  value: prod.openrisknet.org
```

Parameter name

Optional default value

Parameters allow to configure the deployment

Lazar template - DeploymentConfig

```
- kind: DeploymentConfig
 apiVersion: v1
...
 spec:
   ...
   template:
     ...
     spec:
       containers:
       - name: lazar
         image: docker.io/gebele/lazar-rest:${IMAGE_TAG}
         ports:
         - containerPort: 8088
           protocol: TCP
         ...
         imagePullPolicy: Always
```

Container specification

Use of a parameter

```
  readinessProbe:
    httpGet:
      path: "/"
      port: 8088
      scheme: "HTTP"
    failureTreshold: 4
    initialDelaySeconds: 30
    periodSeconds: 30
    timeoutSeconds: 4
  livenessProbe:
    ...
  resources:
    requests:
      cpu: ${CPU_REQUEST}
      memory: ${MEMORY_REQUEST}
    limits:
      cpu: ${CPU_LIMIT}
      memory: ${MEMORY_LIMIT}
```

Readiness and liveness probes

Resource requests and limits

Lazar template - Service and Route

```
- kind: Service
 apiVersion: v1
  metadata:
    name: lazar
    ...
 spec:
   ports:                          Port spec
   - name: lazar
     protocol: TCP
     port: ${LAZAR_SERVICE_PORT}
     targetPort: 8088
     nodePort: 0
   selector:                       Pods to service
     name: lazar
   type: ClusterIP
   sessionAffinity: None
```

```
- kind: Route
 apiVersion: v1
 metadata:
   name: ${ROUTE_NAME}
   annotations:
     kubernetes.io/tls-acme: ${TLS}
 spec:
   host: ${ROUTE_NAME}.${ROUTES_BASENAME}      Public hostname
   to:
     kind: Service
     name: lazar
   tls:
     termination: edge
     insecureEdgeTerminationPolicy: Redirect
```

Lazar Template - Template Service Broker

These annotations are used by the Template Service Broker to allow the template to be deployed easily through the web console - demo coming later

```
metadata:
 name: lazar
 annotations:
   openshift.io/display-name: lazar toxicity prediction service
   openshift.io/provider-display-name: Johannes Gutenberg University Mainz - JGU, in silico toxicology gmbh - IST
   openshift.io/documentation-url: https://github.com/OpenRiskNet/home.git
   openshift.io/support-url: https://github.com/OpenRiskNet/home/issues
   description: lazar (lazy structure-activity relationships) is a modular framework for predictive toxicology.
     Similar to the read across procedure in toxicological risk assessment, lazar creates local
     QSAR (quantitative structure-activity relationship) models for each compound to be predicted.
   iconClass: ''
   tags: lazar,prediction,rest
```

Other ORN app templates

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

Contains templates for most of the ORN partner applications plus some additional 3rd party applications such as JupyterHub.

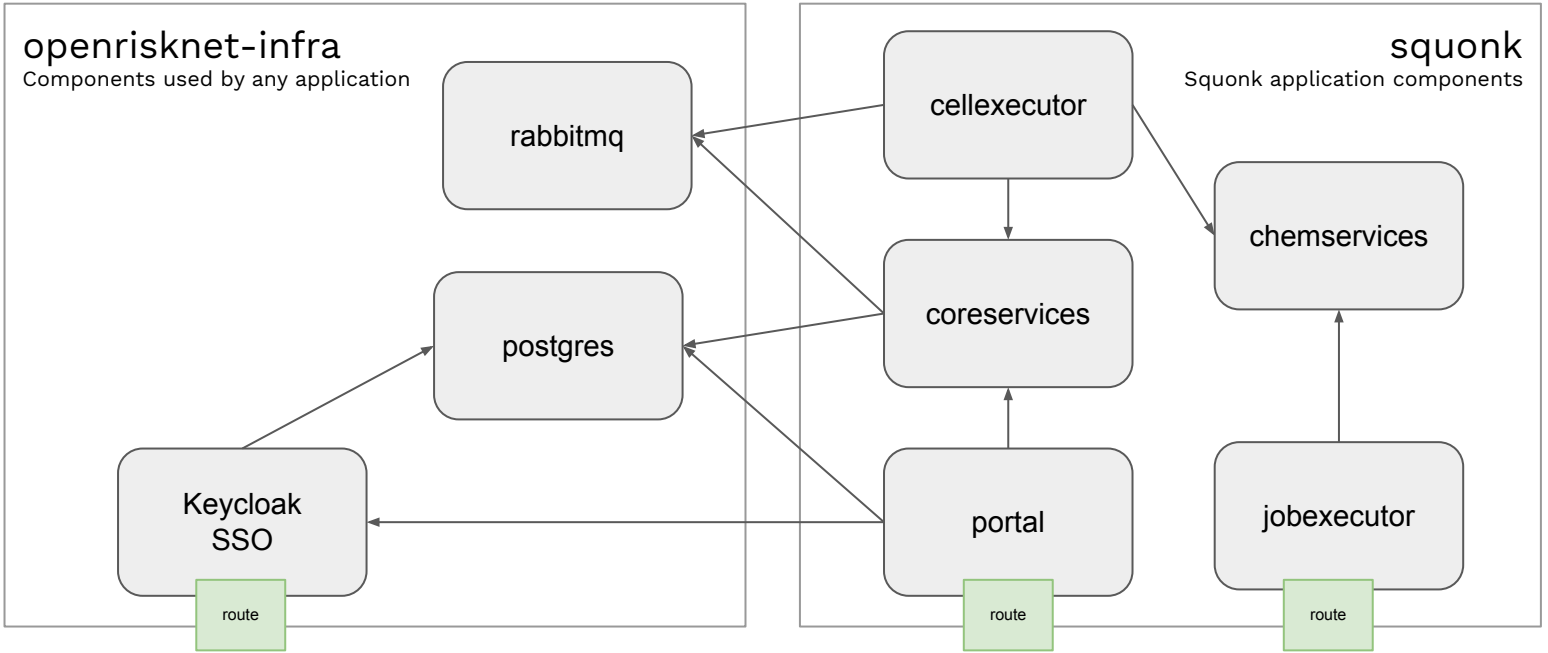
Simple examples: bridgedb, jguweka

More complex examples: jupyterhub, squonk, jaqpot

We'll now look at Squonk as a more complex example.

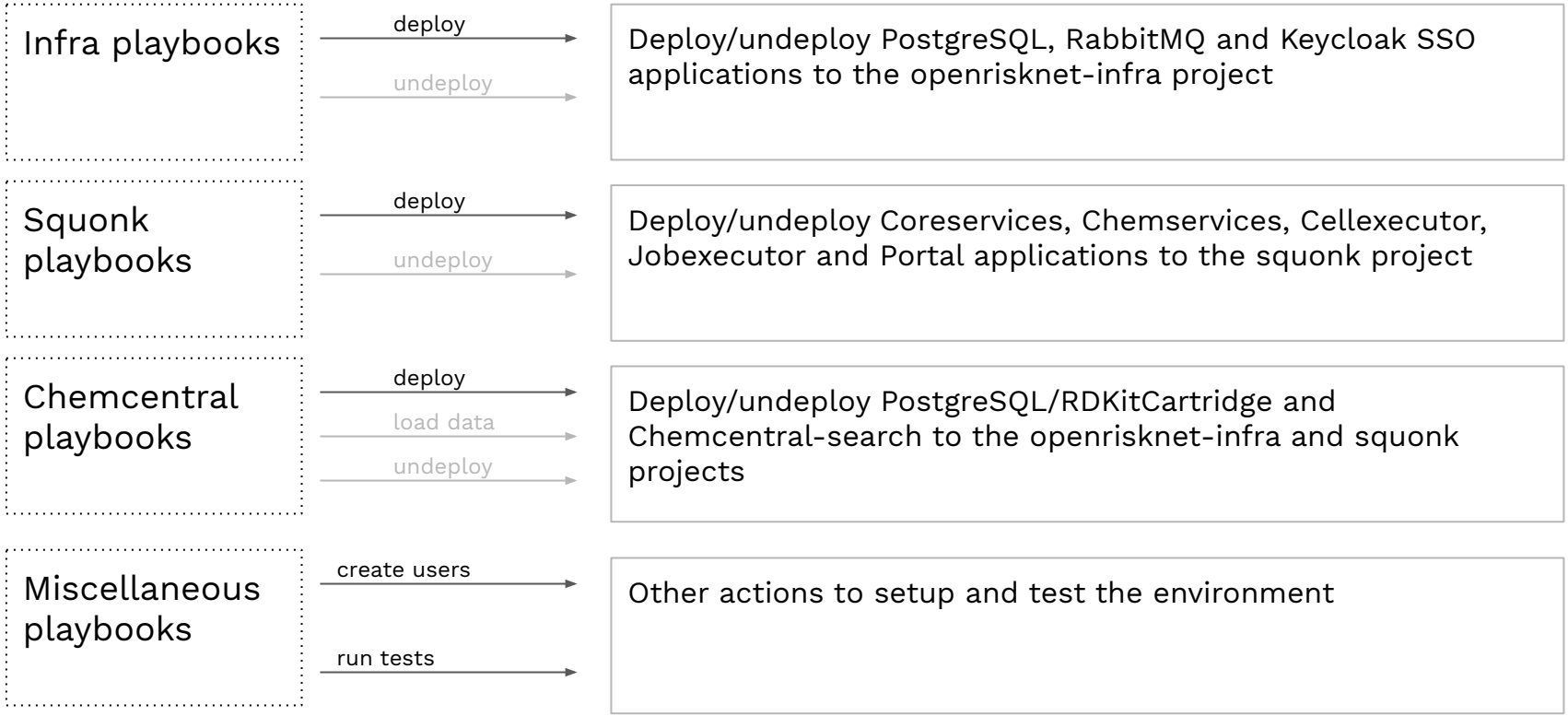
Squonk architecture overview

some of the squonk components



Squonk deployment

https://github.com/InformaticsMatters/squonk/tree/master/openshift/ansible

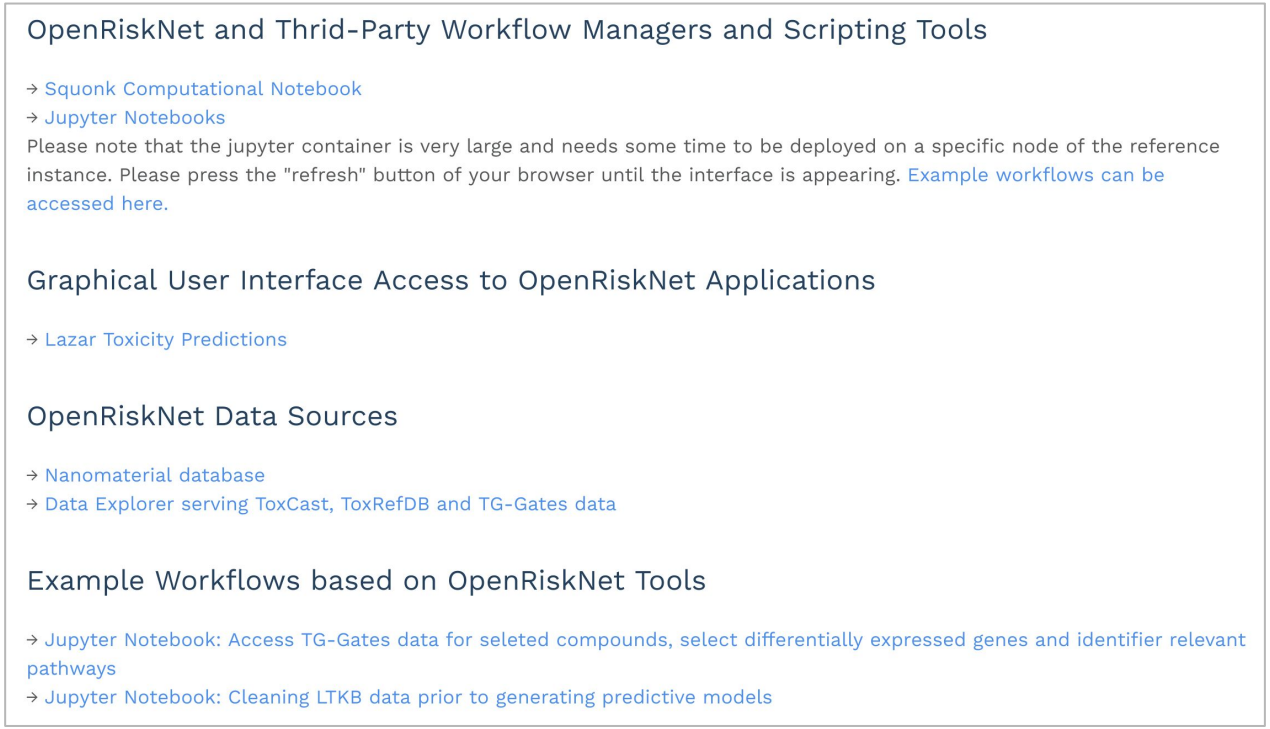


Best practices

- Consider security aspects of your containers
- Try to create small containers
- Consider how much resource (CPU, memory) your containers need
- Use SSO for authentication

Guidelines are provided here:
https://github.com/OpenRiskNet/home/wiki/Deployment-Guidelines

For full details of the ORN partner and 3rd party applications that can be deployed to an ORN VE look here: https://home.prod.openrisknet.org/



OpenRiskNet and Thrid-Party Workflow Managers and Scripting Tools

→ Squonk Computational Notebook
→ Jupyter Notebooks
Please note that the jupyter container is very large and needs some time to be deployed on a specific node of the reference instance. Please press the "refresh" button of your browser until the interface is appearing. Example workflows can be accessed here.

Graphical User Interface Access to OpenRiskNet Applications

→ Lazar Toxicity Predictions

OpenRiskNet Data Sources

→ Nanomaterial database
→ Data Explorer serving ToxCast, ToxRefDB and TG-Gates data

Example Workflows based on OpenRiskNet Tools

→ Jupyter Notebook: Access TG-Gates data for seleted compounds, select differentially expressed genes and identifier relevant pathways
→ Jupyter Notebook: Cleaning LTKB data prior to generating predictive models

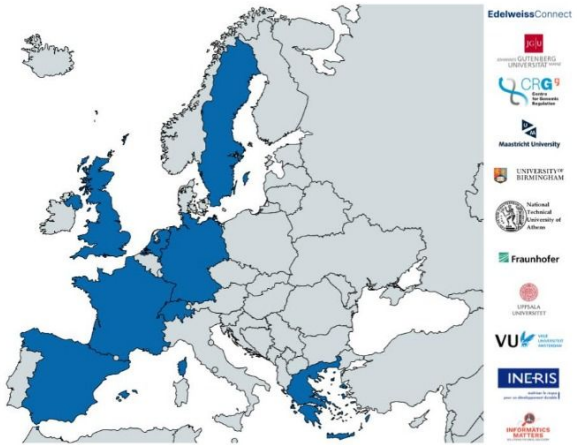
Conclusion

- OpenShift/Kubernetes is a powerful application platform
- A wide range of options for deploying applications
- Support for simple and complex application topologies
- Can also include building applications from source
- Significant learning curve is involved
- But provides excellent approach for robust and automated deployment of applications

Acknowledgements

OpenRiskNet (Grant Agreement 731075) is a project funded by the European Commission within Horizon 2020 Programme

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 INERIS, France (INERIS)
P12 Vrije Universiteit Amsterdam, Netherlands (VU)