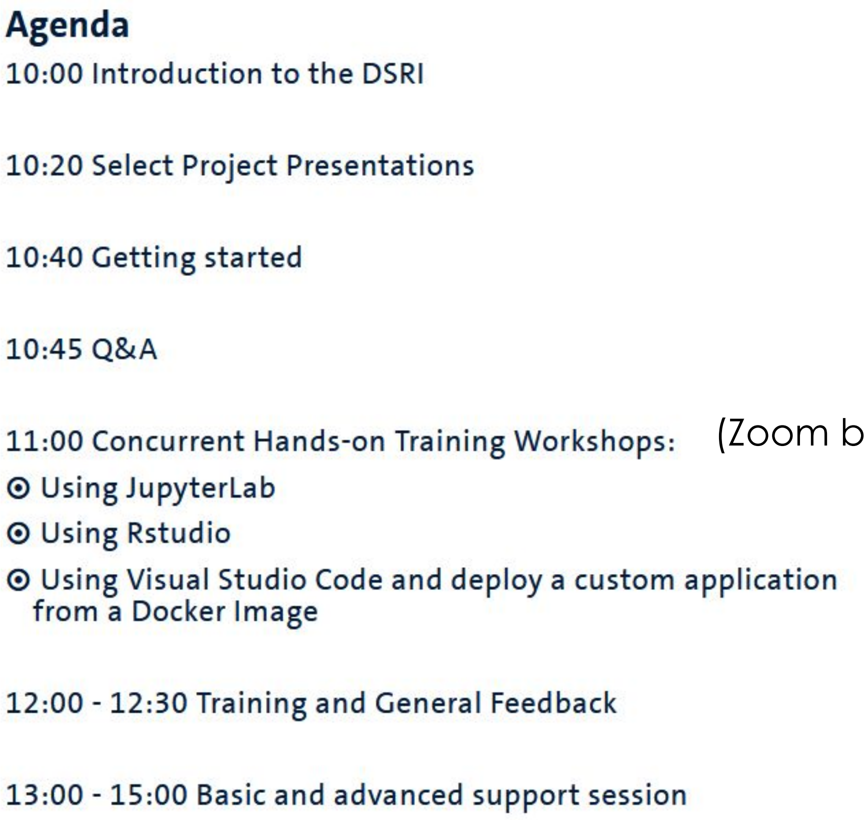
Maastricht University

Institute of Data Science

DSRI Community Event

2nd December 2020

DSRI Community Event



Agenda

10:00 Introduction to the DSRI

10:20 Select Project Presentations

10:40 Getting started

10:45 Q&A

11:00 Concurrent Hands-on Training Workshops:

(Zoom break out rooms)

- Using JupyterLab
- Using Rstudio
- Using Visual Studio Code and deploy a custom application from a Docker Image

12:00 - 12:30 Training and General Feedback

13:00 - 15:00 Basic and advanced support session

DSRI Team

Michel Dumontier
Institute of Data Science
Project Lead

Vincent Emonet
Institute of Data Science
Support

Binosha Weerarathna
Institute of Data Science
User outreach and training

Arjen van Wijngaarden
Fourco
Consultant

Chris Kuipers
ICTS
Linux System Engineer

Marcel Brouwers
ICTS
Linux System Engineer

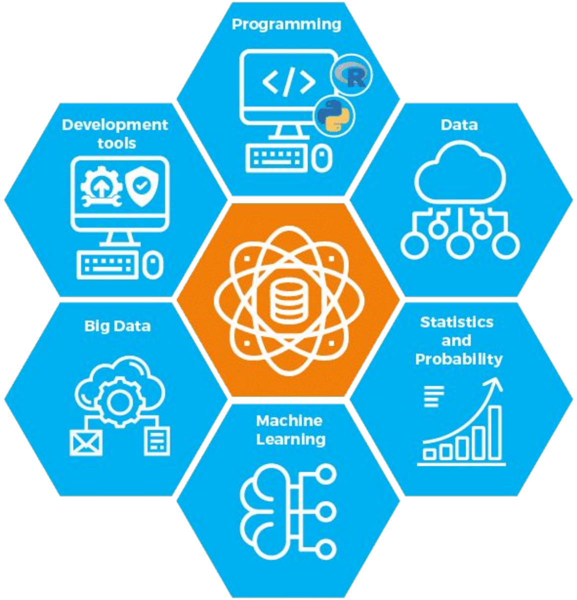
Jordy Frijns
ICTS
Linux System Engineer

Armand Habets
ICTS
Product Manager

Emiel Kremers
Fourco
Consultant

Vision

An <u>effective</u>, <u>scalable</u>, and <u>sustainable</u> data science computing infrastructure at Maastricht University



initiated in 2018 as a collaboration between the Institute of Data Science and ICTS

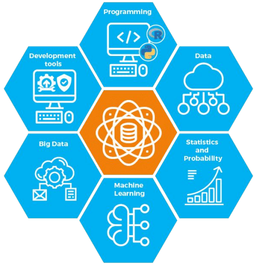
Vision

An effective, scalable, and sustainable data science computing infrastructure at Maastricht University

Effective in that DSRI helps you get data science work done with much less administration

Scalable in both that you can use more resources for your problem, and that we can grow the cluster when needed

Sustainable in that it is an infrastructure that is maintained by its community of users along with the UM



Why is DSRI needed?

1. **Lack of a shared research computing infrastructure** has resulted in *multiple isolated, incompatible, and independently managed infrastructures* that have differing policies and patchy compliance to organizational, national and international regulations, that cannot be combined.
2. **Researchers should focus on their research**, instead of being burdened with administrating computational infrastructure
3. UM wants to make research results **FAIR** - Findable, Accessible, Interoperable, Reusable - a shared infrastructure would foster best practices to help researchers achieve **FAIR and reproducible research and workflows**.
4. A shared infrastructure will enhance the position of the UM and help **attract and retain data science talent**

Design Objectives

An infrastructure that

- **Facilitates large scale data analysis** using big data technologies using both CPU + GPU computing
- Reduces administrative overhead with self-administrative user interfaces
- Enables **component deployment via containers** (Docker)
- Enables **data sharing** via a flexible and shared storage solution
- Is **scalable and fault-tolerant** by combining global monitoring with auto-migration

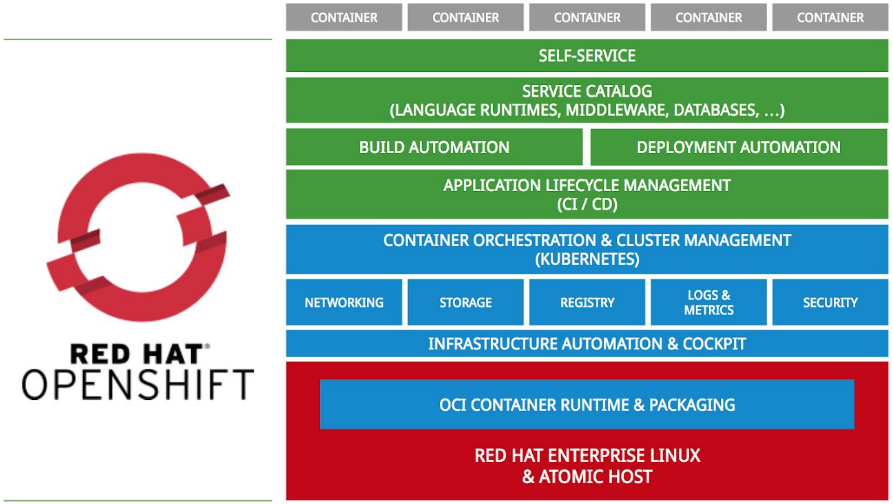
An Orchestrated Solution

Automated configuration, coordination, and management of DSRI

Orchestration using OpenShift and Kubernetes

MAPR platform is used as Hadoop compatible storage

Runs Docker-based containers



CONTAINER | CONTAINER | CONTAINER | CONTAINER | CONTAINER

SELF-SERVICE

SERVICE CATALOG
(LANGUAGE RUNTIMES, MIDDLEWARE, DATABASES, ...)

BUILD AUTOMATION | DEPLOYMENT AUTOMATION

APPLICATION LIFECYCLE MANAGEMENT
(CI / CD)

CONTAINER ORCHESTRATION & CLUSTER MANAGEMENT
(KUBERNETES)

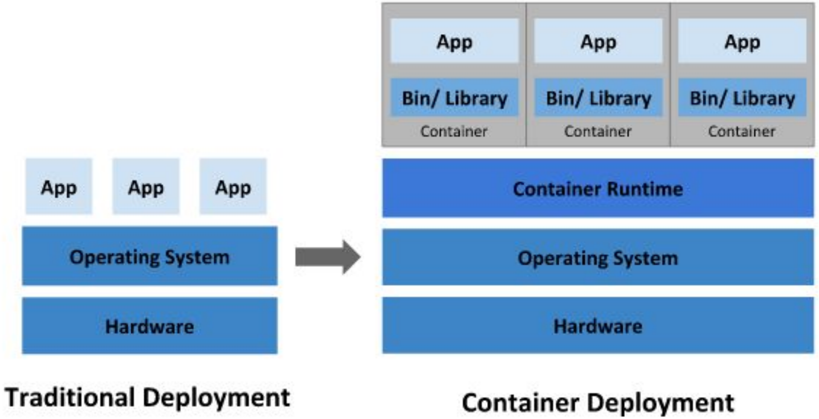
NETWORKING | STORAGE | REGISTRY | LOGS & METRICS | SECURITY

INFRASTRUCTURE AUTOMATION & COCKPIT

OCI CONTAINER RUNTIME & PACKAGING

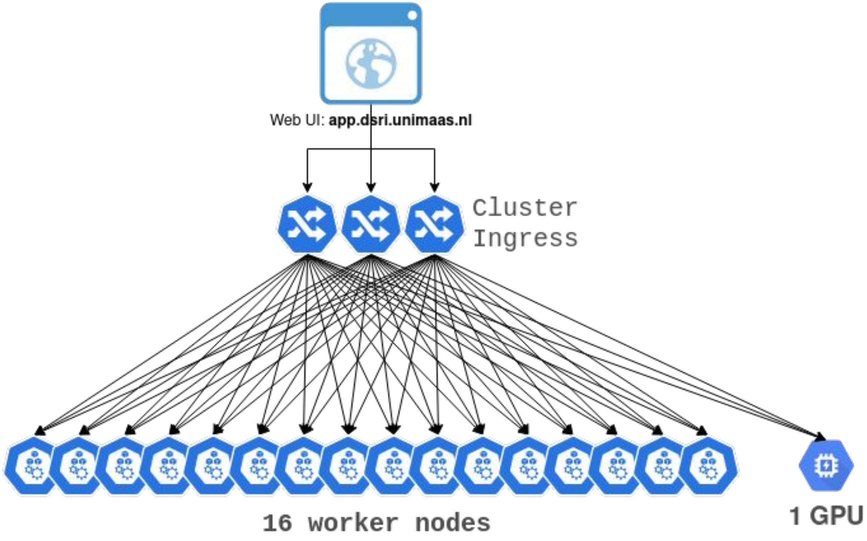
RED HAT ENTERPRISE LINUX
& ATOMIC HOST

Containers have exactly what is needed to deploy an application



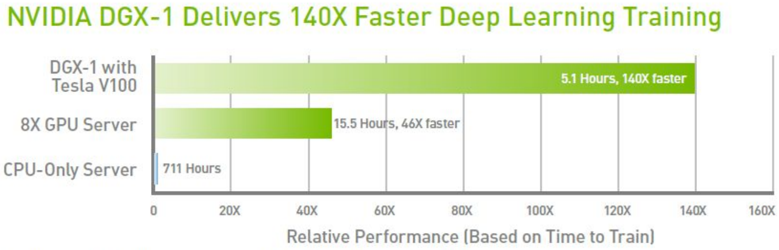
- Applications are prepared with everything that is required to successfully deploy them *elsewhere*
- Cloud and OS portability: runs on Ubuntu, RHEL, on-premises, and in major public clouds
- Higher efficiency in using underlying compute resources through load balancing and scaleout
- Protect underlying systems from application specific exploits
- Easy for users to find and redeploy specific apps for their own use

DSRI configuration



Cluster as of November, 2020

- 16x CPU nodes
  - 2x AMD EPYC 7551
  - 512 GB Memory
  - 120TB (1920TB total)
- 1x GPU node (Nvidia DGX-1)
  - 8x NVIDIA Tesla V100 32 GB/GPU
  - 40,960 Nvidia CUDA cores
  - 5,120 Tensor Cores
- 40 Gb/s interconnects



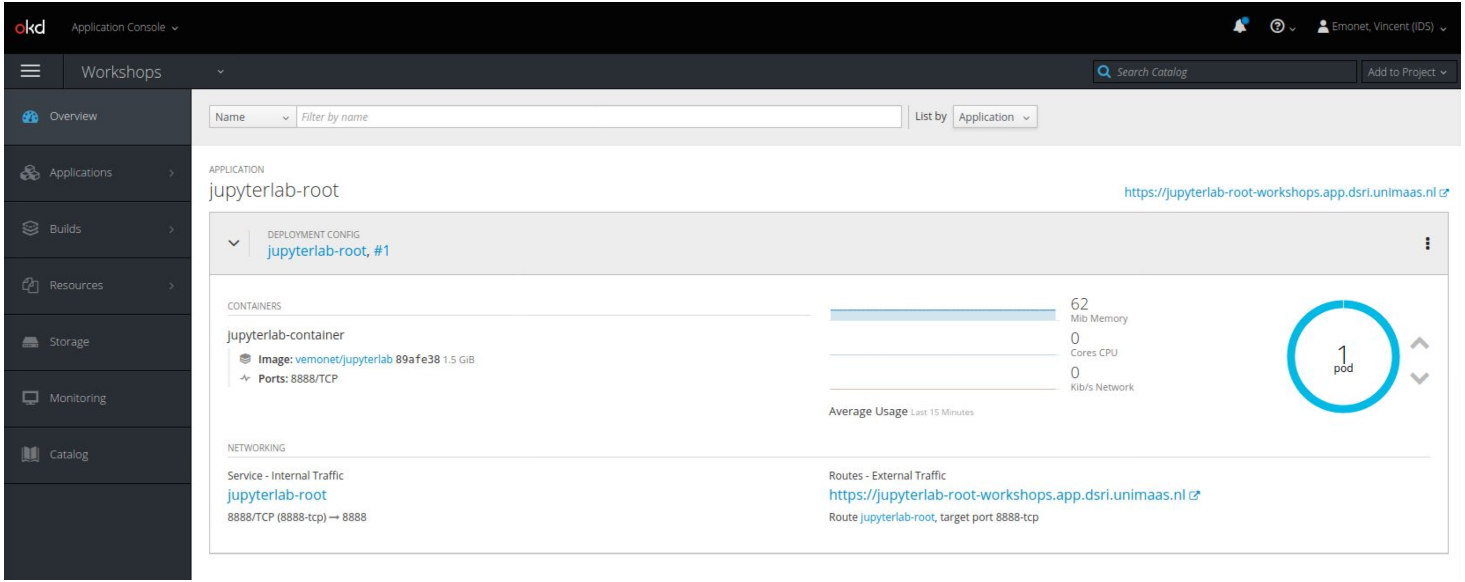
Workload: ResNet-50, 90 epochs to solution | CPU Server: Dual Xeon E5-2699v4, 2.6GHz

What can be done on the DSRI

- Run **Data Science applications** in Docker container 🐳 on the UM network
  - JupyterLab (scipy, tensorflow, all-spark, and more)
  - JupyterHub with GitHub authentication
  - RStudio, with a complementary Shiny server
  - VisualStudio Code server
  - Tensorflow or PyTorch on Nvidia GPU
  - SQL, NoSQL and Graph databases (PostgreSQL, MongoDB, Blazegraph...)
  - Apache Flink cluster for Streaming applications
- You can also deploy **any customized container image** (Docker)

Manage your applications

- Through the OpenShift Web UI (behind the VPN)



APPLICATION
jupyterlab-root
https://jupyterlab-root-workshops.app.dsri.unimaas.nl

DEPLOYMENT CONFIG
jupyterlab-root, #1

CONTAINERS
jupyterlab-container
Image: vemonet/jupyterlab 89afe38 1.5 GiB
Ports: 8888/TCP

62 Mib Memory
0 Cores CPU
0 Kib/s Network
Average Usage Last 15 Minutes

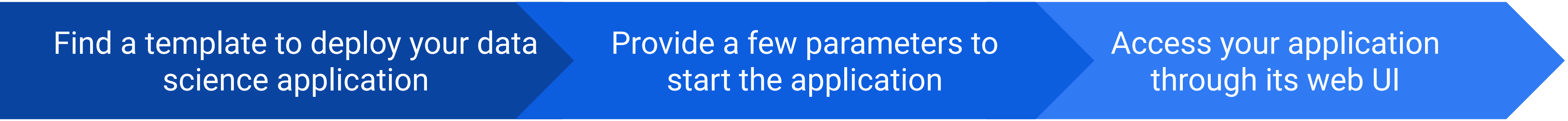
1 pod

NETWORKING
Service - Internal Traffic
jupyterlab-root
8888/TCP (8888-tcp) → 8888

Routes - External Traffic
https://jupyterlab-root-workshops.app.dsri.unimaas.nl
Route jupyterlab-root, target port 8888-tcp

- Or through the terminal using the **oc** command line interface
  - Which is better for some operations, such as loading large datasets

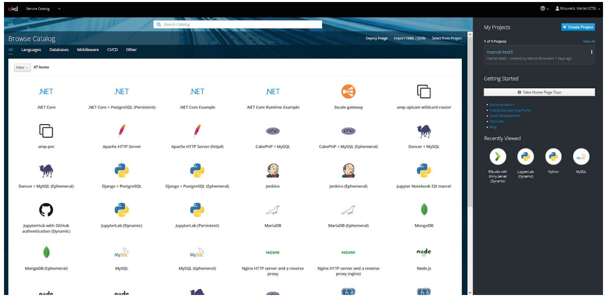
Deploy applications easily using templates

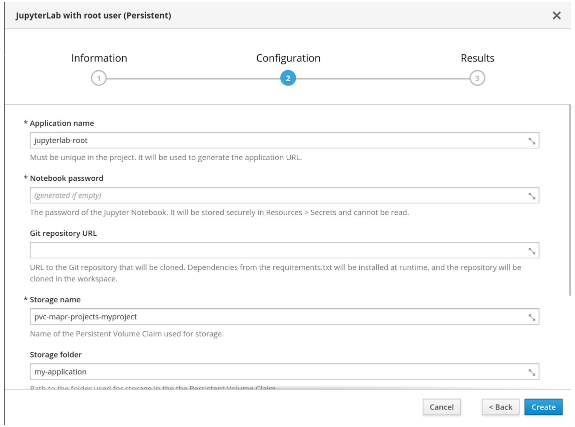

Find a template to deploy your data science application
Provide a few parameters to start the application
Access your application through its web UI

Ask for new templates if needed!

Such as name, password, storage location

Using a URL created by the DSRI
Or connect via the terminal





JupyterLab with root user (Persistent)

* Application name

jupyterlab-root

Must be unique in the project. It will be used to generate the application URL.

* Notebook password

(generated if empty)

The password of the Jupyter Notebook. It will be stored securely in Resources > Secrets and cannot be read.

Git repository URL

URL to the Git repository that will be cloned. Dependencies from the requirements.txt will be installed at runtime, and the repository will be cloned in the workspace.

* Storage name

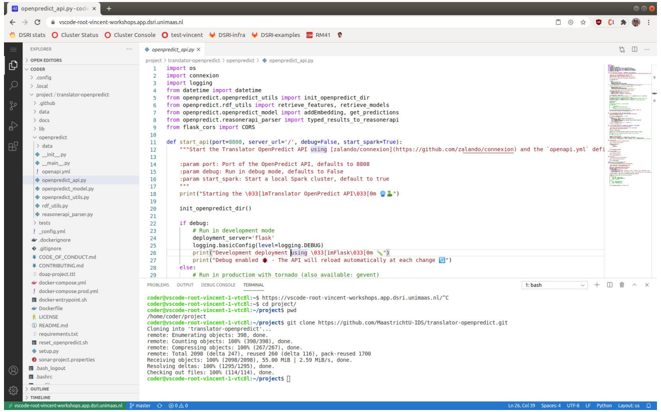
pvc-mapr-projects-myproject

Name of the Persistent Volume Claim used for storage.

Storage folder

my-application

Path to the folder used for storage in the the Persistent Volume Claim.



```python
import os
import connexion
import logging
from datetime import datetime
from openpredict.openpredict_utils import init_openpredict_dir
from openpredict.rdf_utils import retrieve_features, retrieve_models
from openpredict.openpredict_model import addEmbedding, get_predictions
from openpredict.reasonerapi_parser import typed_results_to_reasonerapi
from flask_cors import CORS

def start_api(port=8808, server_url='/', debug=False, start_spark=True):
    """Start the Translator OpenPredict API using [zalando/connexion](https://github.com/zalando/connexion) and the `openapi.yml` defi

    :param port: Port of the OpenPredict API, defaults to 8808
    :param debug: Run in debug mode, defaults to False
    :param start_spark: Start a local Spark cluster, default to true
    """
    print("Starting the \033[1mTranslator OpenPredict API\033[0m 🔮🐍")

    init_openpredict_dir()

    if debug:
        # Run in development mode
        deployment_server='flask'
        logging.basicConfig(level=logging.DEBUG)
        print("Development deployment using \033[1mFlask\033[0m 🧪")
        print("Debug enabled 🐞 - The API will reload automatically at each change 🔃")
    else:
        # Run in production with tornado (also available: gevent)
```

```
coder@vscode-root-vincent-1-vtc8l:~$ https://vscode-root-vincent-workshops.app.dsri.unimaas.nl/^C
coder@vscode-root-vincent-1-vtc8l:~$ cd project/
coder@vscode-root-vincent-1-vtc8l:~/project$ pwd
/home/coder/project
coder@vscode-root-vincent-1-vtc8l:~/project$ git clone https://github.com/MaastrichtU-IDS/translator-openpredict.git
Cloning into 'translator-openpredict'...
remote: Enumerating objects: 398, done.
remote: Counting objects: 100% (398/398), done.
remote: Compressing objects: 100% (267/267), done.
remote: Total 2098 (delta 247), reused 260 (delta 116), pack-reused 1700
Receiving objects: 100% (2098/2098), 55.00 MiB | 2.59 MiB/s, done.
Resolving deltas: 100% (1295/1295), done.
Checking out files: 100% (114/114), done.
coder@vscode-root-vincent-1-vtc8l:~/project$ 
```

Or define your application deployment!

- Any Docker image can be deployed on the DSRI with a "bit" of configuration
  - In short, you will need to write some YAML files to define the different how to deploy your app (port, storage, resources limitations, etc)
- The DSRI supports **Helm**, the package manager for Kubernetes
  - To deploy existing deployments
  - Or create new deployments with multiple services easily

DSRI Storage Solutions

Data will not be lost when pod get restarted

Ephemeral Storage

- Storage is bound to the pod
- Data will be lost when the pod is deleted
- We do not propose this solution anymore, feel free to ask us if you need it

Dynamic storage

- Automatically create when starting an application
- Can also be created in the OpenShift web UI
- Does not work with container using the root user

Persistent storage

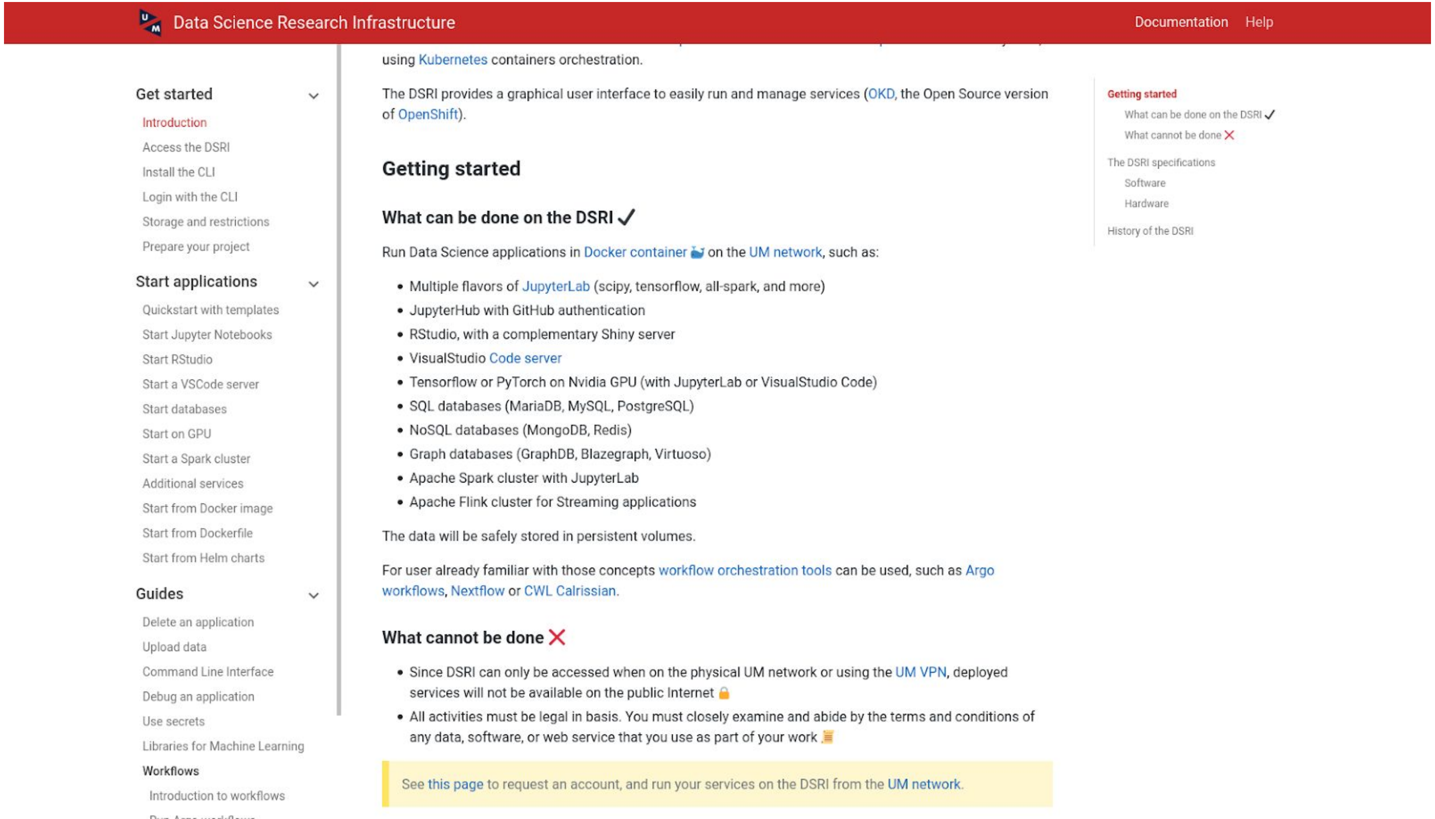
- Can be created by the DSRI team
- Data will not be lost when pod get restarted.

Reasons to use the DSRI

- **Run** your work on a **remote server** at UM through popular web UI (Jupyter notebooks, RStudio, VisualStudio Code) instead of your computer
- **Get faster results** with 120 cores to parallelize tasks, or the 500GB memory to run large workloads
- Make use of **best practices** (using git to version and share code) and provide shared environments (containers) to improve project FAIRness
- **Develop and share** these results with your (UM) collaborators

Collaborative documentation website

https://maastrichtu-ids.github.io/dsri-documentation



using Kubernetes containers orchestration.

The DSRI provides a graphical user interface to easily run and manage services (OKD, the Open Source version of OpenShift).

Getting started

What can be done on the DSRI ✓

Run Data Science applications in Docker container 🐳 on the UM network, such as:

- Multiple flavors of JupyterLab (scipy, tensorflow, all-spark, and more)
- JupyterHub with GitHub authentication
- RStudio, with a complementary Shiny server
- VisualStudio Code server
- Tensorflow or PyTorch on Nvidia GPU (with JupyterLab or VisualStudio Code)
- SQL databases (MariaDB, MySQL, PostgreSQL)
- NoSQL databases (MongoDB, Redis)
- Graph databases (GraphDB, Blazegraph, Virtuoso)
- Apache Spark cluster with JupyterLab
- Apache Flink cluster for Streaming applications

The data will be safely stored in persistent volumes.

For user already familiar with those concepts workflow orchestration tools can be used, such as Argo workflows, Nextflow or CWL Calrissian.

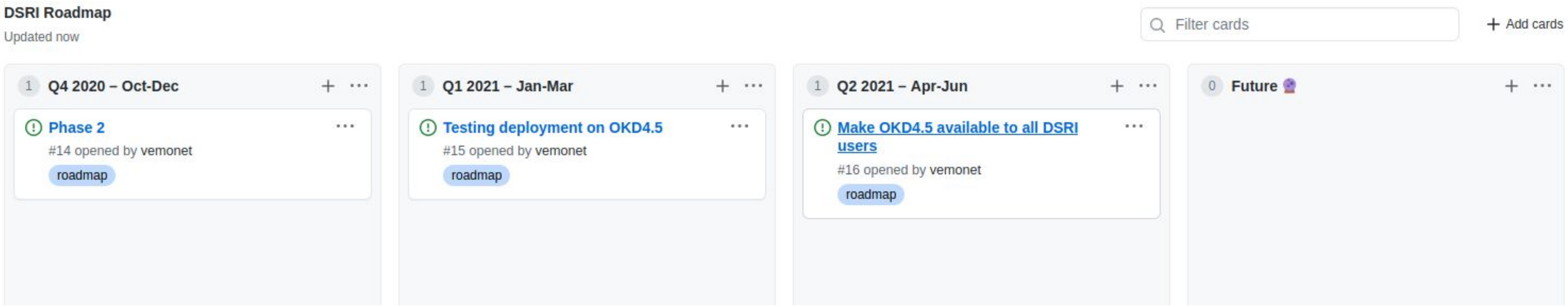
What cannot be done ✗

- Since DSRI can only be accessed when on the physical UM network or using the UM VPN, deployed services will not be available on the public Internet 🔒
- All activities must be legal in basis. You must closely examine and abide by the terms and conditions of any data, software, or web service that you use as part of your work 📜

> See this page to request an account, and run your services on the DSRI from the UM network.

User Community

- We use slack as instant messaging platform for DSRI communications
  - Get the invitation to Slack after registering to the DSRI
  - **#helpdesk** channel
- Issues tracker on GitHub
  - https://github.com/MaastrichtU-IDS/dsri-documentation/issues
- A public roadmap for the DSRI
  - https://github.com/MaastrichtU-IDS/dsri-documentation/projects/1



DSRI Roadmap

Updated now

Filter cards

+ Add cards

1 **Q4 2020 – Oct-Dec**

Phase 2
#14 opened by vemonet
roadmap

1 **Q1 2021 – Jan-Mar**

Testing deployment on OKD4.5
#15 opened by vemonet
roadmap

1 **Q2 2021 – Apr-Jun**

Make OKD4.5 available to all DSRI users
#16 opened by vemonet
roadmap

0 **Future** 🔮

67 registered users and 50 documented projects

bigcat	Department of Bioinformatics	FHML
fhml	Faculty of Health, Medicine and Life Sciences	FHML
hsr	Department of Health Services Research	FHML
maastro	Maastro Clinic	FHML
NUTRIM	School of Nutrition and Translational Research in Metabolism	FHML
phartox	Department of Pharmacology & Toxicology	FHML
pn	Department of Psychiatry and Neuropsychology	FHML
tgx	Department of Toxicogenomics	FHML
Tech Lab	Law and Tech Lab	FL
dke	Department of Data Science and Knowledge Engineering	FSE
fse	Faculty of Science and Engineering	FSE
gwfp	Gravitational Waves and Fundamental Physics	FSE
ids	Institute of Data Science	FSE
lofse	LO-FSE	FSE
macsbio	MACSBIO System Biology	FSE
msp	Maastricht Science Program	FSE
MSCM		SBE
sbe	School of Business and Economics	SBE
icts	ICT services	UM
um	Maastricht University	UM

learning
python
data
using
Analysis
statistical
machine
processing
models
project
R
use
JAVA
deep
Classify
GWAS
websites
Neural
quantum
omics
MRI
RStudio
predict
model
Text
network
tensorflow
NLP
running
development
identify
developed
develop
pytorch
users
support
techniques
advanced
estimation
software
case
Trying
aims
primarily
large
field
multiple
algorithms
content
store
specific
analyses
extract
Study
sound
lots
etc

Wordcloud from project descriptions

Project presentation

CBCT to CT translation for Adaptive Radiotherapy

What are our future plans?

- **A vibrant community-supported infrastructure**
  - Weekly technical meetings and monthly planning meetings
  - Advice and feedback from new advisory board
  - Regular (2-3x annual) community meetings and training workshops
  - Improved user experience and multi-media documentation
  - Mon-Fri user support
- **Infrastructure improvements**
  - testing OKD 4.5 on a subset of the cluster + CEPH storage (ongoing)
  - resource scheduling and quota management (GPU, CPU)
  - security, data protection, and disaster recovery policies
- **Deploy new Data Science and Machine Learning platforms**
  - Apache Spark, OpenDataHub, KubeFlow, FAIRscape
  - Public-facing applications by the UM research community
- Develop **community-based governance and policies;** invite new investors, secure long term financing, and grain external funding.

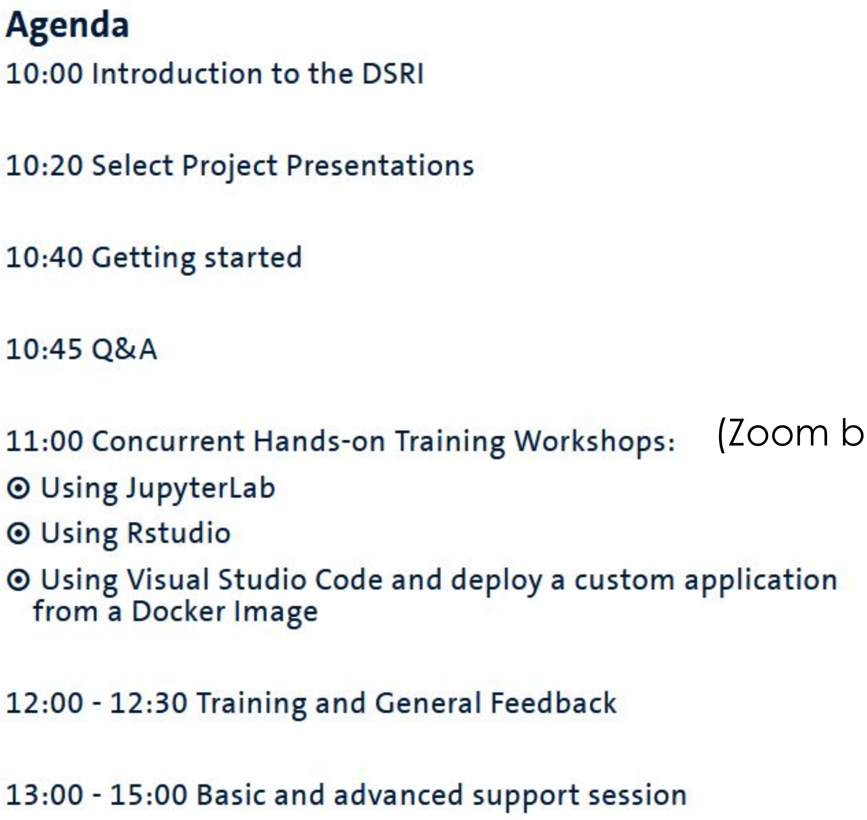
Workshop: Start an Application

Go to your breakout room

And follow the workshop instructions at

https://maastrichtu-ids.github.io/dsri-workshop-start-app

DSRI Community Event



Agenda

10:00 Introduction to the DSRI

10:20 Select Project Presentations

10:40 Getting started

10:45 Q&A

11:00 Concurrent Hands-on Training Workshops:

- Using JupyterLab
- Using Rstudio
- Using Visual Studio Code and deploy a custom application from a Docker Image

12:00 - 12:30 Training and General Feedback

13:00 - 15:00 Basic and advanced support session

(Zoom break out rooms)

Feedback

Share your thoughts on your first experience with the DSRI

1. What did you think about DSRI getting started and setup procedure?
2. What other applications would you like to see on the DSRI?
3. What would take it to get you starting to use DSRI (more?)

Questions?