ETH zürich

Pipeline Interoperability Using Containers

in the Context of Personalized Health Research

2017 hpc-ch Forum – Containers for HPC

Balazs Laurenczy

ETH Zürich – Scientific IT Services

Outline

- Motivation
- Pipeline Interoperability

  e-Science Coordination team

  pilot project

- EnhanceR project

  follow up to the pilot project

Outline

- Motivation
- Pipeline Interoperability

  e-Science Coordination Team

  pilot project

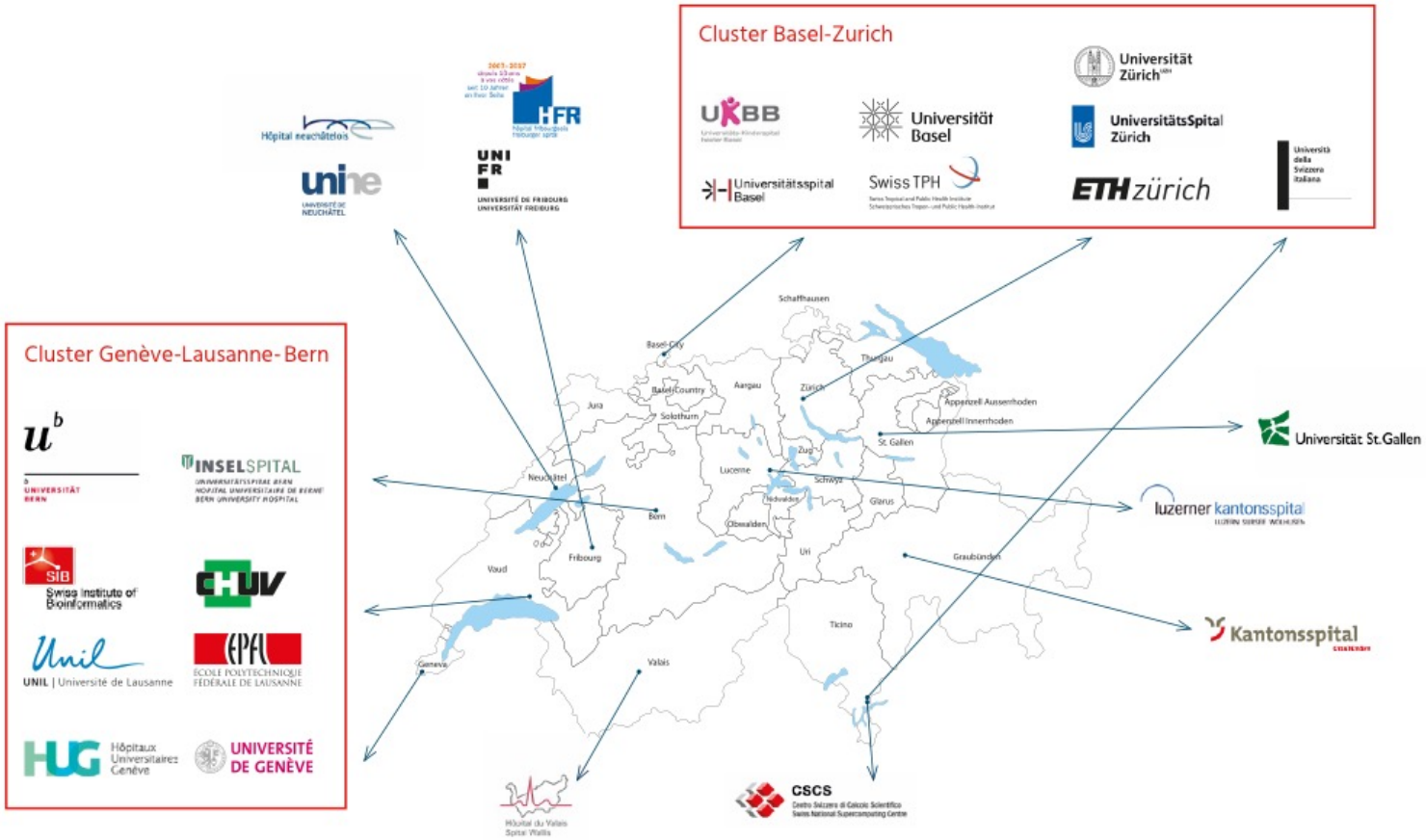
- EnhanceR project

  follow up to the pilot project

Motivation – SPHN

SPHN Swiss Personalized Health Network

"… nationally coordinated data infrastructure ensuring **data interoperability** of local and regional information systems with special emphasis on clinical data management systems enabling **effective exchange of patient data** (e.g. disease phenotypes)."

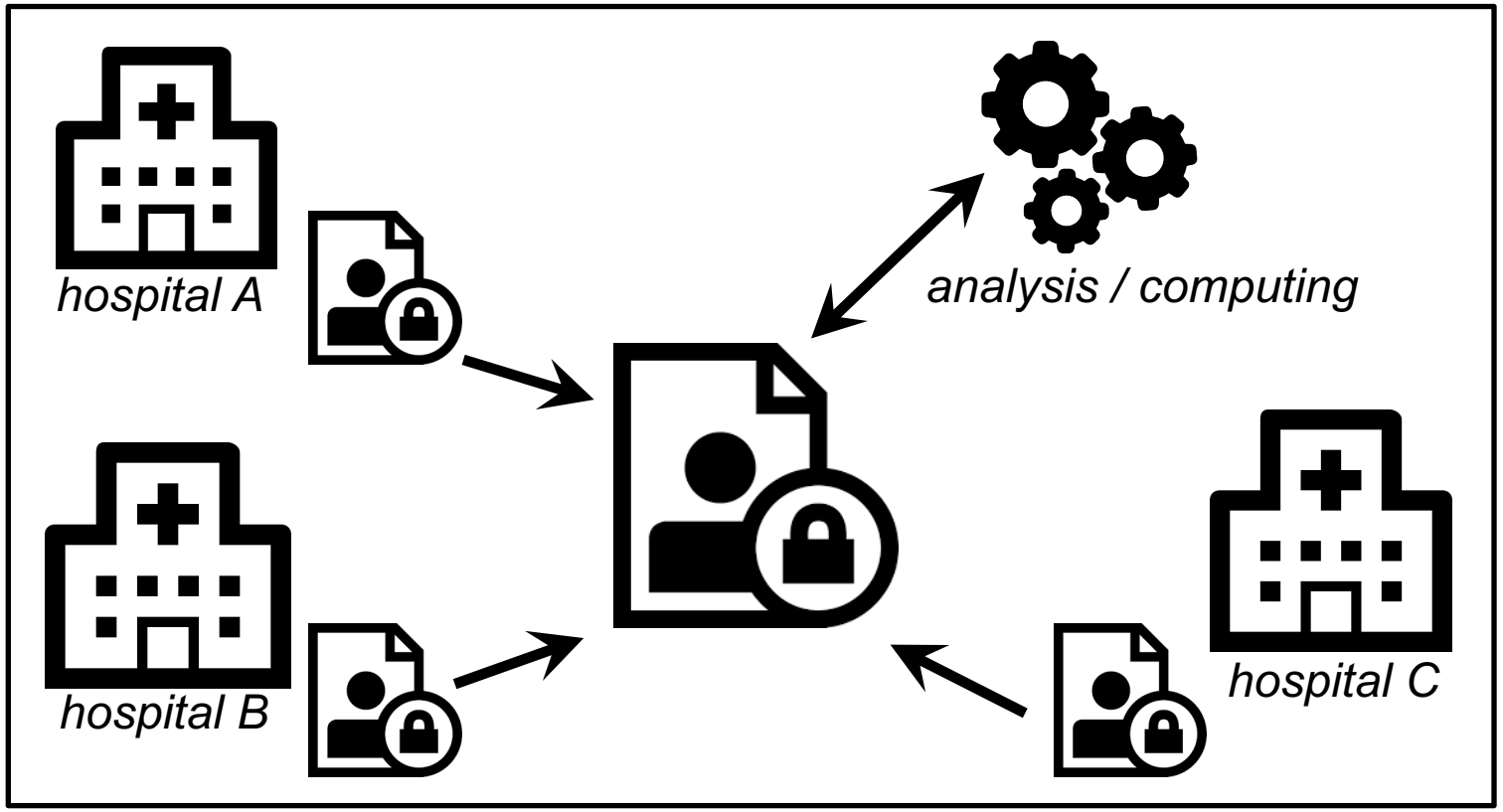


source: https://www.sphn.ch/en/about.html

Motivation – Requirements from Personalized Health Data

Swiss Personalized Health Network

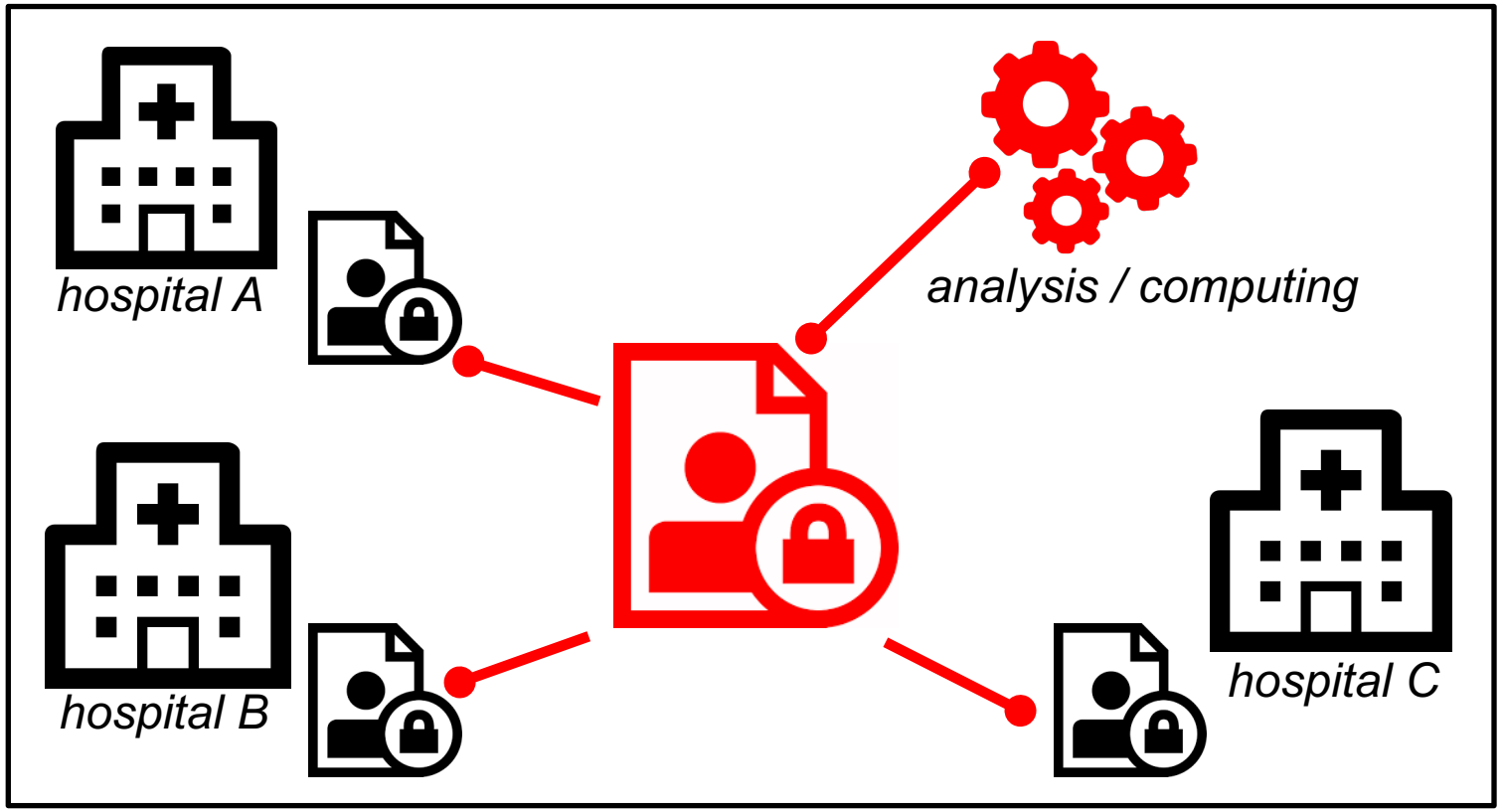
Swiss **P**ersonalized **H**ealth **N**etwork



Motivation – Requirements from Personalized Health Data

Swiss Personalized Health Network

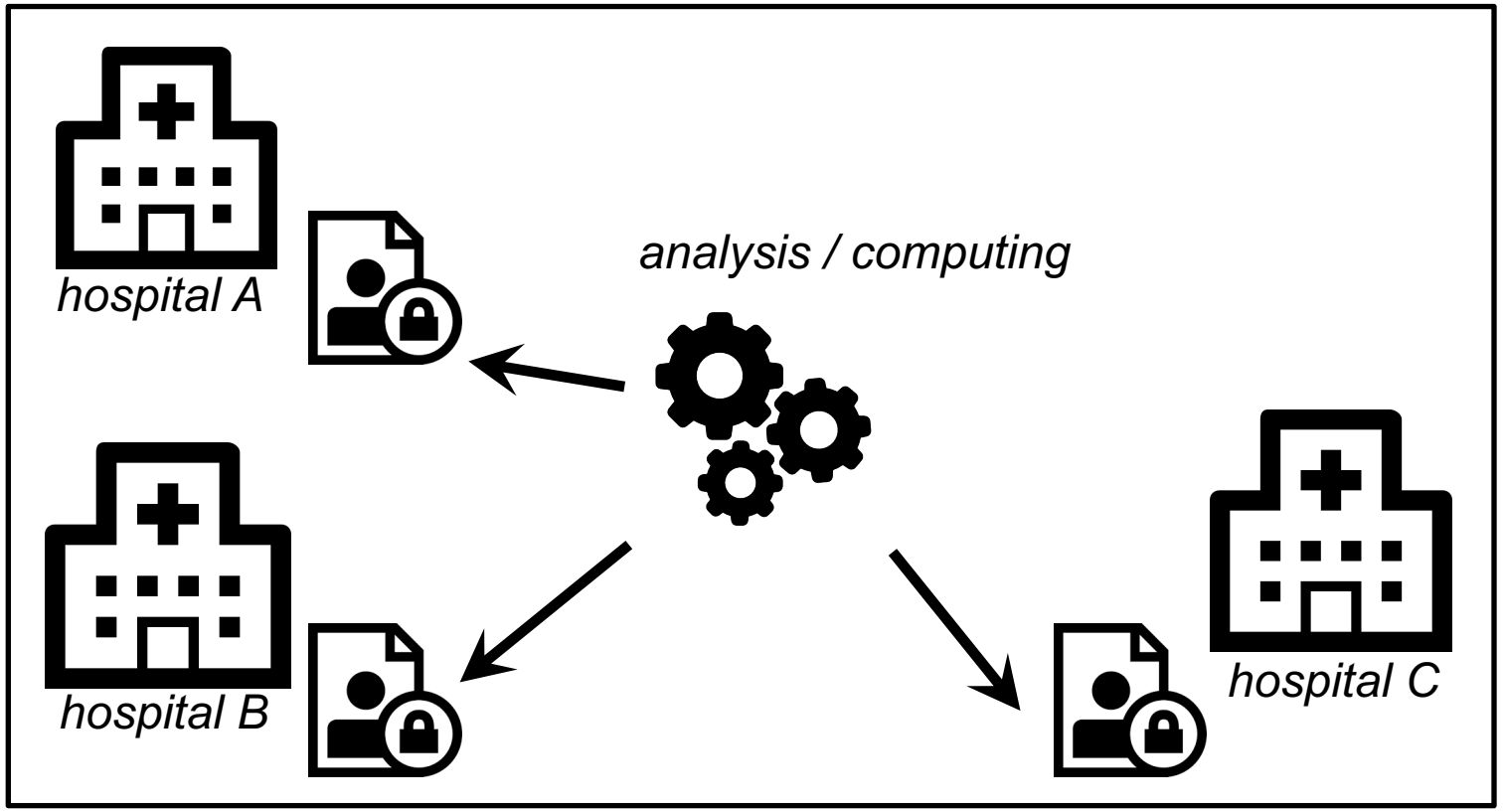
Swiss Personalized Health Network



Motivation – Requirements from Personalized Health Data

Swiss Personalized Health Network

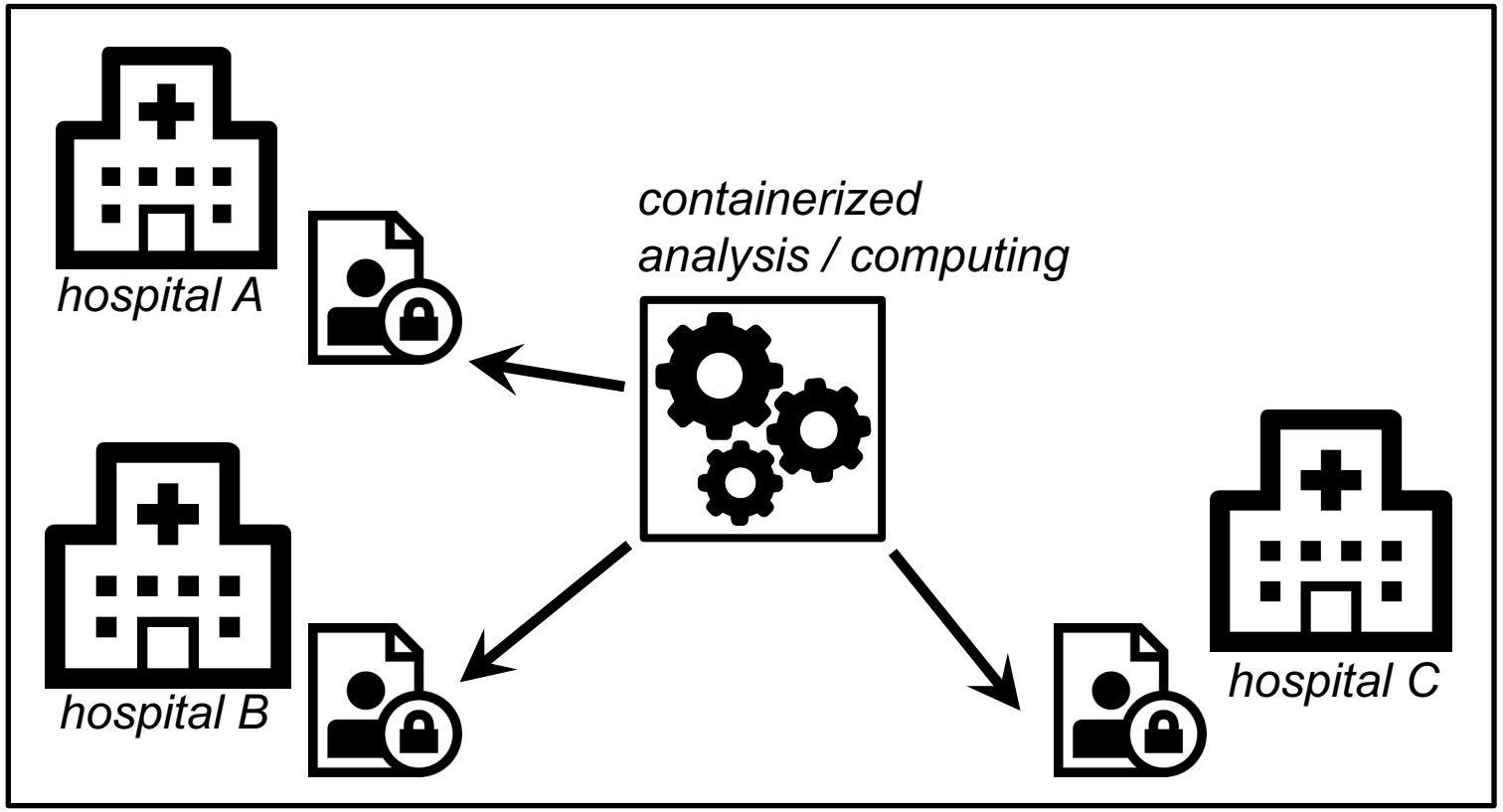
Swiss **P**ersonalized **H**ealth **N**etwork



Motivation – Requirements from Personalized Health Data

Swiss Personalized Health Network

Swiss **P**ersonalized **H**ealth **N**etwork



Motivation – Requirements from Personalized Health Data

Swiss **P**ersonalized **H**ealth **N**etwork

- Sensitive medical data exist in many **different location**
- **Computing goes to the data**, not vice versa
- Hard requirement of SPHN to **share the pipelines** with other SPHN members

- Improve the **sharing of software** within the Swiss Scientific IT Services via the usage of **containers**
  - Need for a *framework* to fetch, validate, deploy and run containers
  - Helps to lower the effort to make a pipeline available on a different infrastructure (no need to install dependencies, etc.)

Outline

- Motivation
- Pipeline Interoperability

  e-Science Coordination Team

  pilot project

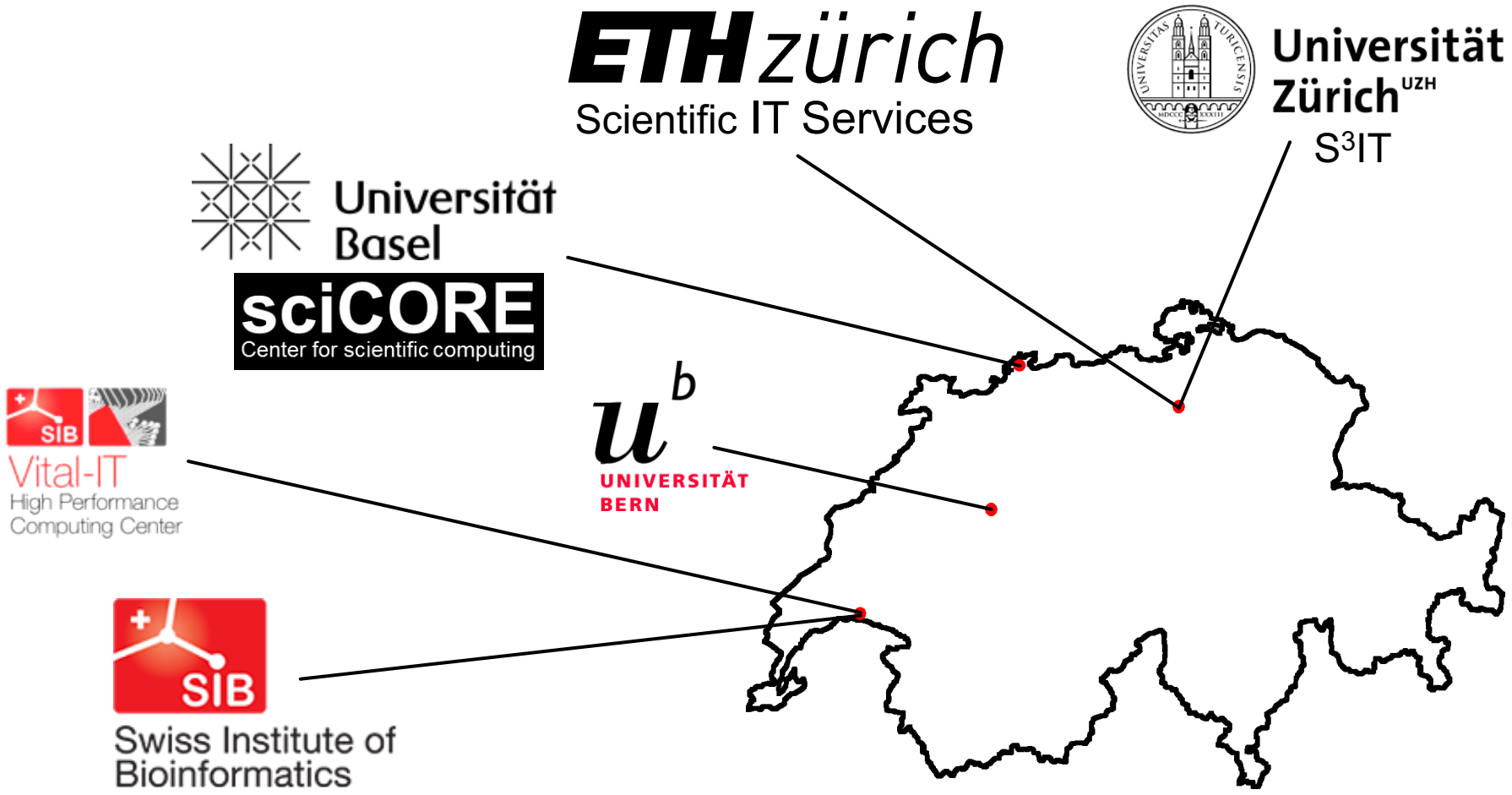
- EnhanceR project

  follow up to the pilot project

Pipeline Interoperability project – Goals

- Assess **pipeline interoperability** between the different **HPC clusters** of Switzerland using **container** technologies
  - *Pipeline:* chain of data-processing software (e.g. bioinformatics workflow)
  - *Pipeline interoperability:* the possibility to run the same pipeline on different physical infrastructure with the same outcome / output
  - *Container:* OS-level virtualization method for running software without launching an entire VM
- Challenge of using containers on an **HPC infrastructure**
  - Docker and Singularity
  - Interaction with scheduler
- Create a **community** of container-developers / -deployers / -maintainers / -users

Pipeline Interoperability project – Community



Pipeline Interoperability project – Summary & Outcomes

- **Technology** used:

Docker — most used software container platform – unavailable on HPC environment due to high privilege requirement

Singularity — HPC-friendly container platform that has compatibility with Docker

- **Milestones** (completed by all members of the community)
  - Singularity installation and *basic* “hello world” pipeline running
  - More *complex* pipeline (bioinformatics oriented)
  - Docker → Singularity conversions via different means (Dockerfile, Dockerhub, own registry, etc.)

singularity-pipeline python runner module

- **Guidelines** document (*http://bit.ly/guidelines_containers*)
  - gathers know-how and best practices
  - defines standards for achieving *pipeline interoperability*

Pipeline Interoperability project – Milestones

- Singularity **installed**

Extracted from the guidelines document

```
./configure
make
sudo make install

$PREFIX/etc/singularity/singularity.conf
```

`mount home = yes` Defines if the user home folder should be available inside the container by default.

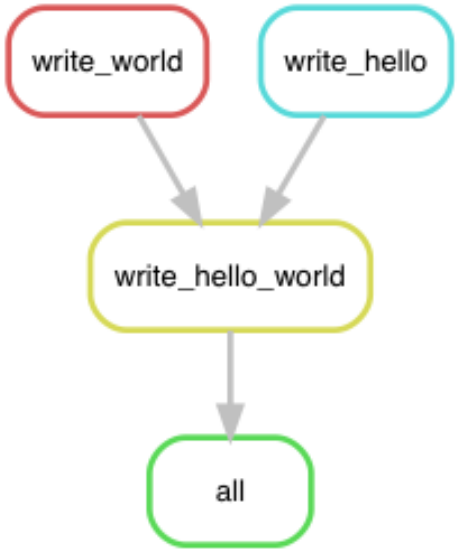
`mount tmp = yes` Defines if the `/tmp` folder from the host should be available inside the container by default.

`bind path` Define multiple paths from the host system which should be available inside the container by default. A typical use case is to configure the cluster's shared scratch file system or the cluster's projects file system to be available inside the containers.

`enable overlay = yes` This option is very interesting as it enables the use of the use of the overlay file system (*overlayfs*) so that paths which do not exists inside the container are created on the fly. For example, …

Pipeline Interoperability project – Milestones

- *Singularity installed*
- “hello world pipeline” **building**



hello world pipeline

“hello world pipeline” Dockerfile

```
FROM ubuntu:latest

RUN apt-get -y update
RUN apt-get install -y python3 python3-pip
RUN pip3 install --upgrade pip
RUN pip3 install snakemake

RUN mkdir /pipeline
RUN mkdir /data

ENTRYPOINT ["snakemake", "-q", "-s",
    " /pipeline/hello_world_pipeline.snake", "-d", "/data"]
```

```
# build the Docker image using the "Dockerfile"
sudo docker build -t hello_world_pipeline .

# convert the Docker image to a Singularity image
sudo docker run -v /var/run/docker.sock:/var/run/docker.sock -v
$(pwd):/output -- privileged -t --rm singularityware/docker2singularity
hello_world_pipeline
```

Pipeline Interoperability project – Milestones

- *Singularity installed*
- *“hello world pipeline” building*
- ”hello world pipeline” **running**

“hello world pipeline” Dockerfile

```
FROM ubuntu:latest

RUN apt-get -y update
RUN apt-get install -y python3 python3-pip
RUN pip3 install --upgrade pip
RUN pip3 install snakemake

RUN mkdir /pipeline
RUN mkdir /data

ENTRYPOINT ["snakemake", "-q", "-s",
    " /pipeline/hello_world_pipeline.snake", "-d", "/data"]
```

```
$ singularity run -B ./pipeline:/pipeline -B ./data:/data hello_world_pipeline.img
#2017-06-30 15:17:39,892 | INFO     | hello_world_pipeline.snake: Starting Snakemake pipeline ...
#2017-06-30 15:17:40,004 | INFO     | hello_world_pipeline.snake: Starting Snakemake pipeline ...
#2017-06-30 15:17:40,006 | INFO     | hello_world_pipeline.snake: Pipeline completed.
#2017-06-30 15:17:40,022 | INFO     | hello_world_pipeline.snake: Pipeline completed successfully.
#2017-06-30 15:17:40,034 | INFO     | hello_world_pipeline.snake: Pipeline is valid.
```

Pipeline Interoperability project – Milestones

- ”bioinformatics pipeline” **building**

“bioinformatics pipeline” Dockerfile

```
FROM centos:7.3.1611

RUN echo "Here we are installing software and other dependencies for the container!"
RUN echo "Installing STAR"
RUN yum -y install make gcc gcc-c++ zlib-devel
RUN curl -o /usr/local/src/STAR-2.5.2b.tar.gz https://codeload.github.com/alexdobin/STAR/tar.gz/2.5.2b
RUN cd /usr/local/src && tar xf STAR-2.5.2b.tar.gz
RUN cd /usr/local/src/STAR-2.5.2b/source && make STAR && make STARlong
RUN cp /usr/local/src/STAR-2.5.2b/source/{STAR,STARlong} /usr/local/bin

RUN echo "Installing HTSeq and Pysam"
RUN yum -y install epel-release
RUN yum -y install make gcc gcc-c++ zlib-devel python-devel python-pip numpy
RUN pip install HTSeq==0.6.1p1
RUN pip install Pysam==0.9.0
RUN yum clean all

RUN mkdir /data

ENTRYPOINT ["echo", "-e", "This container includes the following apps:\\n\
    STAR version 2.5.2b - https://github.com/alexdobin/STAR\\n\
    HTSeq version 0.6.1p1 - http://www-huber.embl.de/HTSeq/\\n\
    Pysam version 0.9.0 - https://github.com/pysam-developers/pysam\\n\
    To execute a binary inside the container do \"singularity exec /path/to/container.img binary-name\""]
```

Pipeline Interoperability project – Milestones

- *”bioinformatics pipeline” building*
- ”bioinformatics pipeline” **running**

“bioinformatics pipeline” bash script

```
# step 1. Index the reference genome using the STAR utility
singularity exec scicore_pipeline1.img STAR --runThreadN 4 --outTmpDir ./star_tmp \
    --runMode genomeGenerate --genomeDir ./scicore-pipeline1-input-data/GRCm38_Genome/ \
    --genomeFastaFiles ./scicore-pipeline1-input-data/GRCm38_Genome/Ensembl_GRCm38.fa \
    --sjdbGTFfile ./scicore-pipeline1-input-data/GRCm38_Genome/Ensembl_GRCm38_genes.gtf \
    --outFileNamePrefix=output/STAR/

# remove temporary directory
rm -rf star_tmp

# step 2. Align reads from fastq to indexed reference genome using STAR utility
singularity exec scicore_pipeline1.img STAR --runThreadN 4 --outTmpDir ./star_tmp \
    --readFilesCommand zcat --genomeDir ./scicore-pipeline1-input-data/GRCm38_Genome/ \
    --outSAMstrandField intronMotif --outSAMtype BAM SortedByCoordinate \
    --outReadsUnmapped Fastx --outFileNamePrefix output/STAR/TAM3_ \
    --readFilesIn ./scicore-pipeline1-input-data/fastq/SRR3180540_pass_1.fastq.gz

# step 3. Summarize alignment BAM file to counts table by gene feature using HTSeq utility
singularity exec scicore_pipeline1.img htseq-count -f bam -r pos -s no \
    output/STAR/TAM3_Aligned.sortedByCoord.out.bam \
    ./scicore-pipeline1-input-data/GRCm38_Genome/Ensembl_GRCm38_genes.gtf > output/htseq-count/TAM3.counts
```

```
$ singularity exec [CONTAINER_FILE_NAME] [TOOL_INSIDE THE CONTAINER] [PARAMETERS_FOR_TOOL]
```

Pipeline Interoperability project – Milestones

- Docker -> Singularity conversion via **Dockerfile**

- Docker -> Singularity conversion via **DockerHub**

- Docker -> Singularity conversion via alternative **Docker registry (GitLab)**

```
# build the Docker image using the "Dockerfile"
sudo docker build -t hello_world_pipeline .

# convert the Docker image to a Singularity image
sudo docker run -v /var/run/docker.sock:/var/run/docker.sock
-v $(pwd):/output --privileged -t --rm
singularityware/docker2singularity hello_world_pipeline


# pull from DockerHub
singularity pull docker://blaurenczy/esct-pipeline-
interoperability:hello_world


# pull from SIS "private" repository
singularity pull --size 1024
docker://sissource.ethz.ch:5005/balazsl/esct-pipeline-
interoperability:hello_world
```

Pipeline Interoperability project – Runner module

- **Runner module**
  singularity-pipeline

» Package Index > singularity-pipeline > 0.1

singularity-pipeline 0.1

A runner script for pipelines using Singularity containers

Download
singularity-pipeline-0.1.tar.gz

Not Logged In
Login
Register
Lost Login?
Login with OpenID
Login with Google

Status
Nothing to report

File	Type	Py Version	Uploaded on	Size
singularity-pipeline-0.1.tar.gz (md5)	Source		2017-07-20	4KB

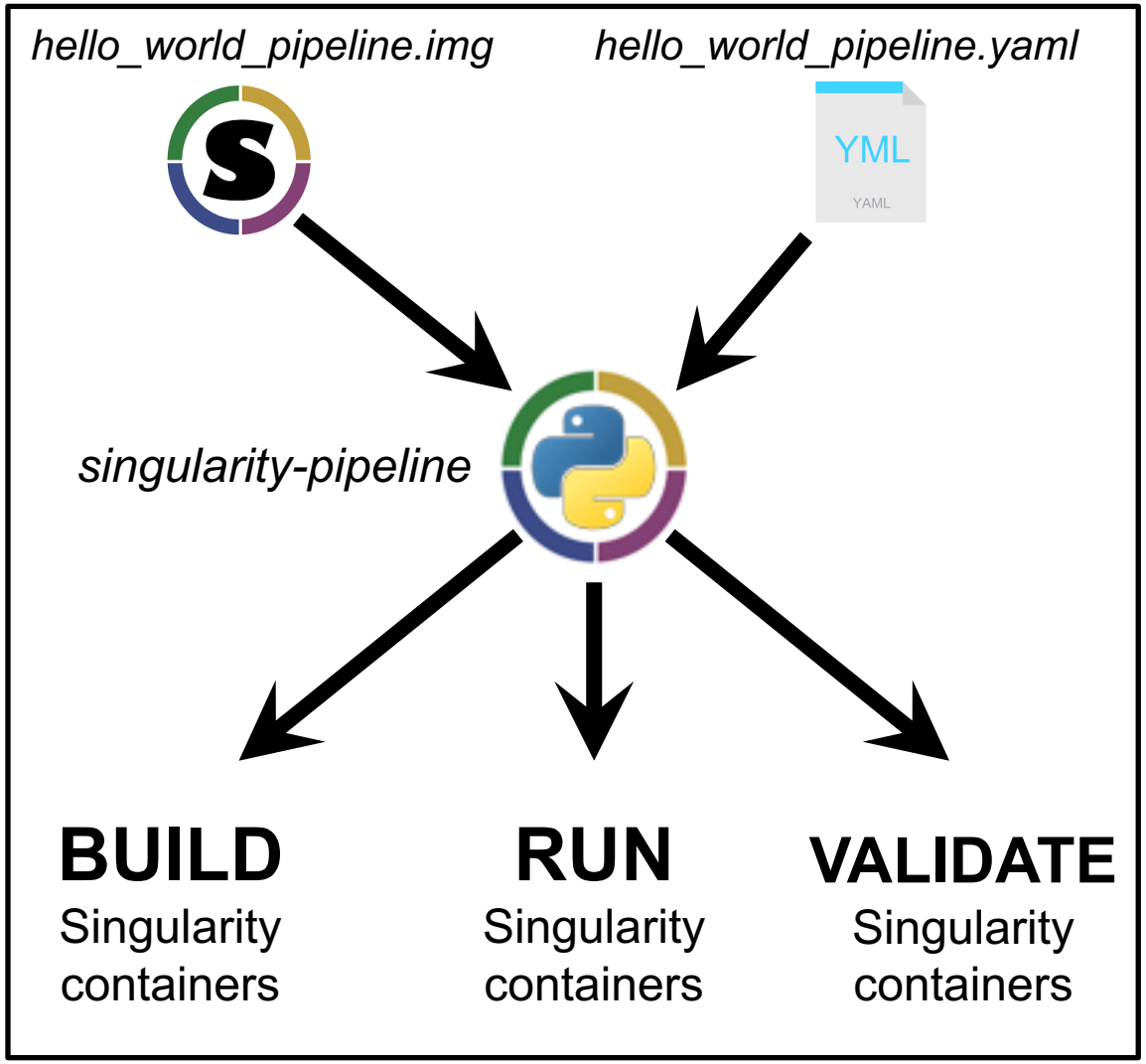
Author: Alexander Kashev
Home Page: https://c4science.ch/diffusion/2915/browse/master/UniBe/
Keywords: singularity container runner
License: MIT
Categories
Development Status :: 3 - Alpha
Programming Language :: Python :: 2
Programming Language :: Python :: 3
Package Index Owner: kav2k
DOAP record: singularity-pipeline-0.1.xml

Alexander Kashev
Faculty Science IT Support
Universität Bern
u^b

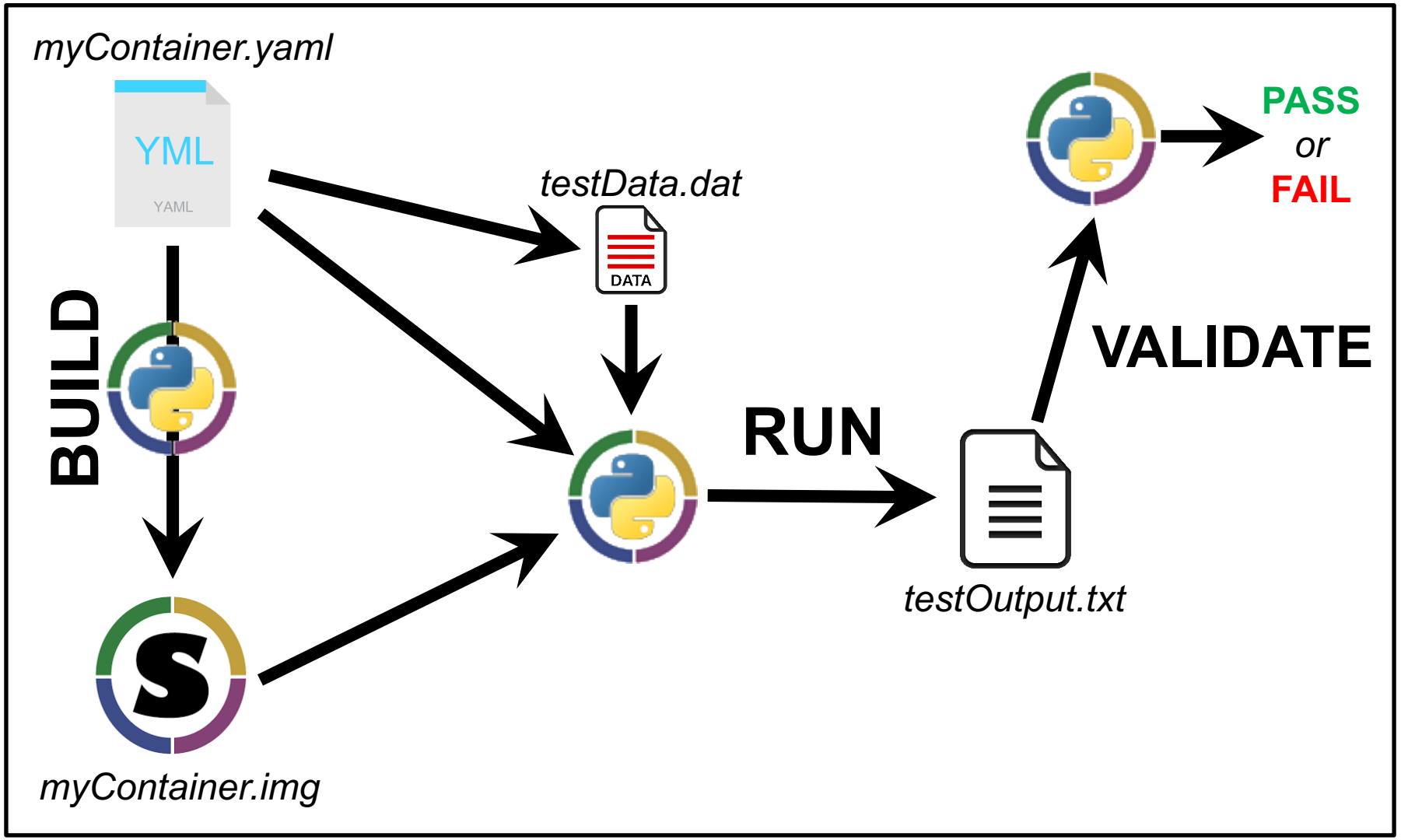
Pipeline Interoperability project – Runner module

"hello world pipeline" YAML description file

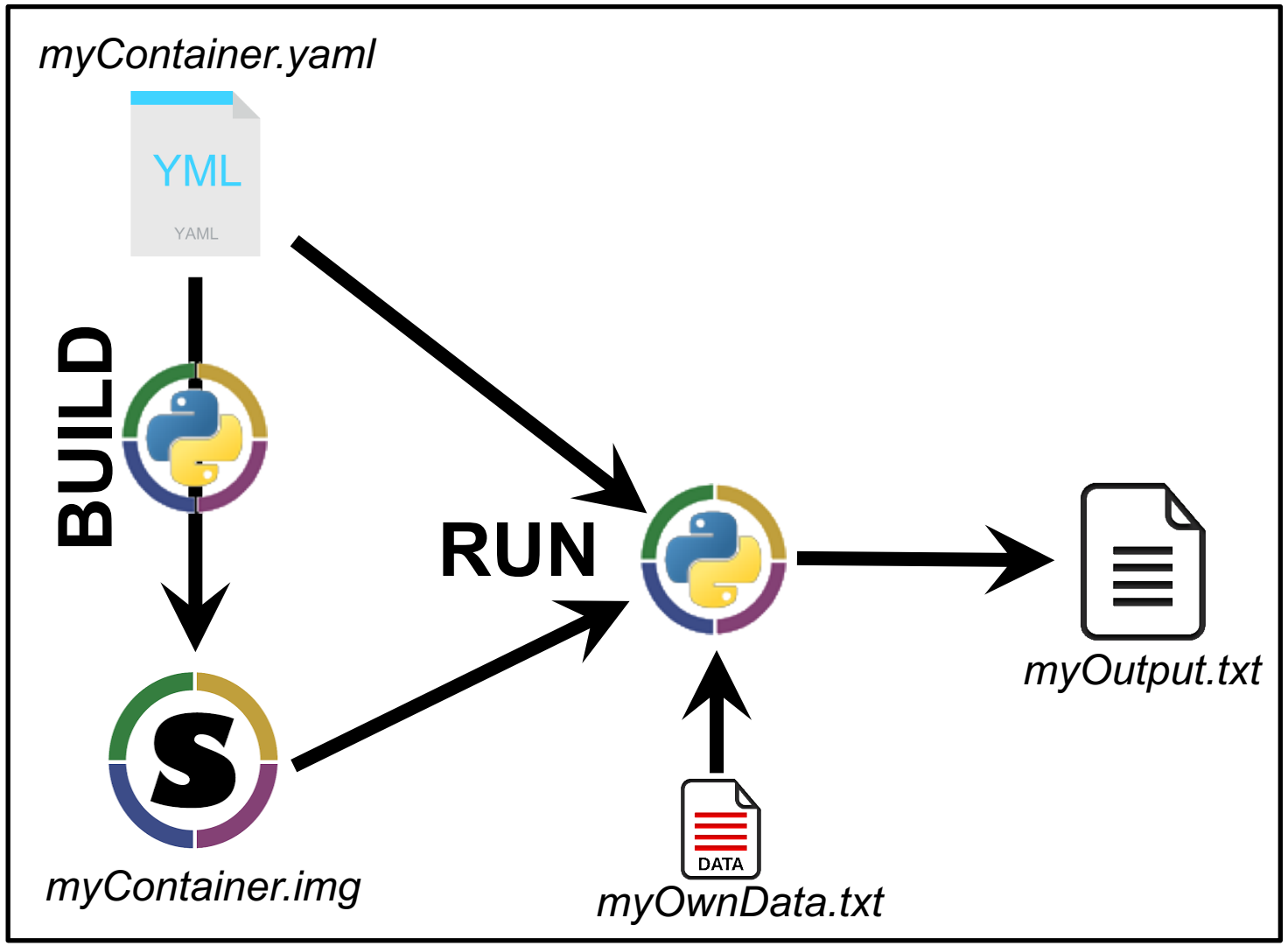
```yaml
name: hello_world_pipeline
version: 1
author: Balazs Laurenczy
author_org: ETHZ
substitutions:
  name: "hello_world_pipeline"
binds:
  - "/usr/lib64:/usr/lib64"
build:
  type: pull
  source: docker://blaurenczy/hello_world:latest
run:
  commands:
    - "{exec} python3 test.py > {name}.out 2> {name}.err"
test:
  validate_commands:
    - "[[ \"$(md5sum {name}.out | cut -f1 -d ' ')\" \
      == \"06c19b37d27dfda293492a4459fe3bc3\" ]] && echo 0"
```



Pipeline Interoperability project – Runner module



Pipeline Interoperability project – Runner module




```
$ singularity-pipeline build

$ singularity-pipeline test

$ singularity-pipeline run
```

ETH zürich

Pipeline Interoperability project – Guidelines

http://bit.ly/guidelines_containers

Guidelines for pipeline interoperability using containers

Status: 27 July 2017

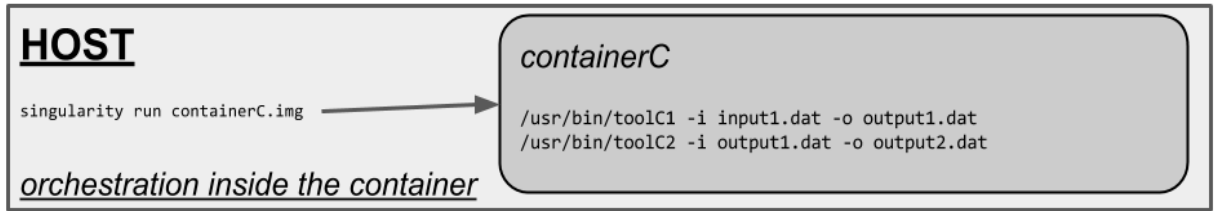
Authors

Balazs Laurenczy (SIS / ETHZ)
Alexander Kashev (ScITS / UniBe)
Séverine Duvaud (SIB)
Vassilios Ioannidis (SIB)
Heinz Stockinger (SIB)
Pablo Escobar López (sciCORE / UniBas)
Sergio Maffioletti (S3IT / UZH)

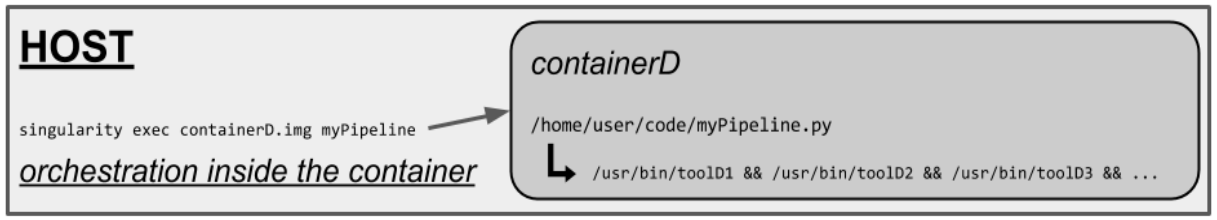
Purpose of this document

The purpose of this document is to report on the eSCT Pipeline Interoperability project, as well as to provide guidelines to reproduce the important milestones achieved in this project. By following these guidelines, an institution should be able to achieve the same level of *interoperability* as all the other project members. This document also contains a description of the know-how acquired and some best practices.

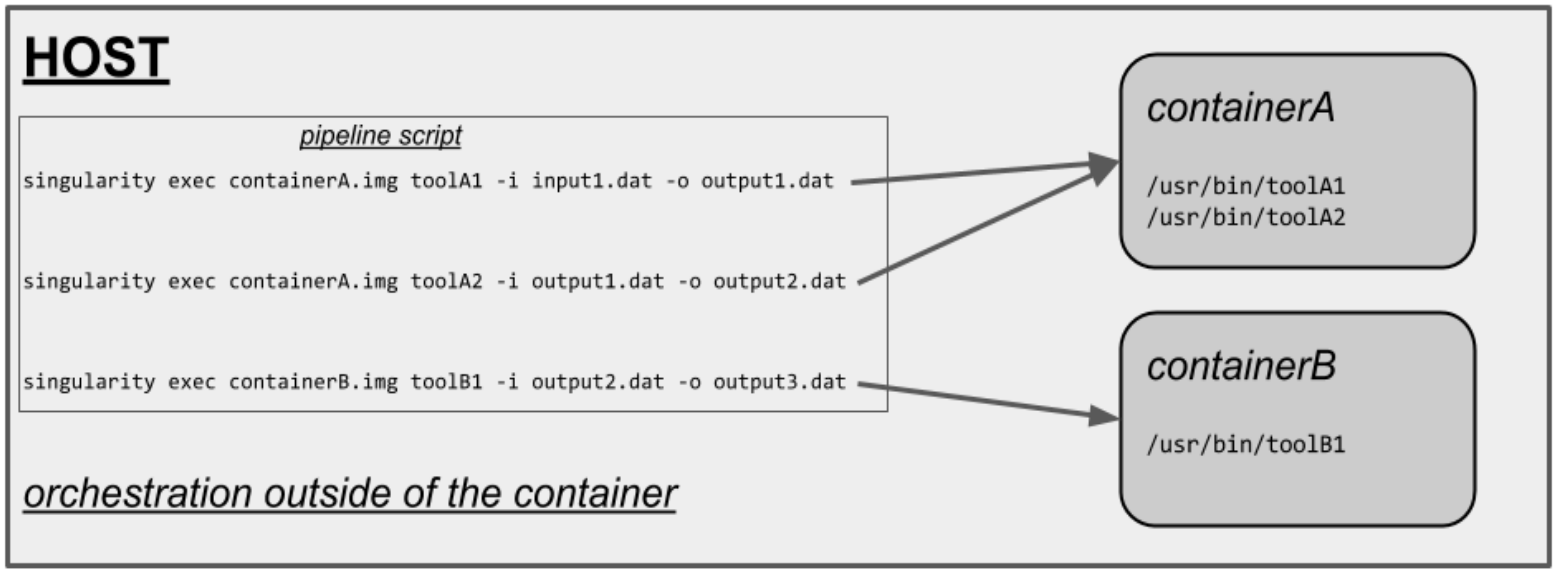
Pipeline Interoperability project – Orchestration strategies



package the whole pipeline inside the container and "just run it"

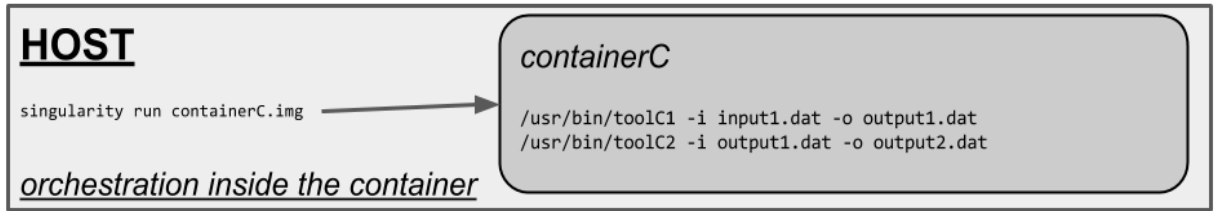


package the whole pipeline inside the container and execute it in a more flexible way (execute only parts of it, change options, etc.)

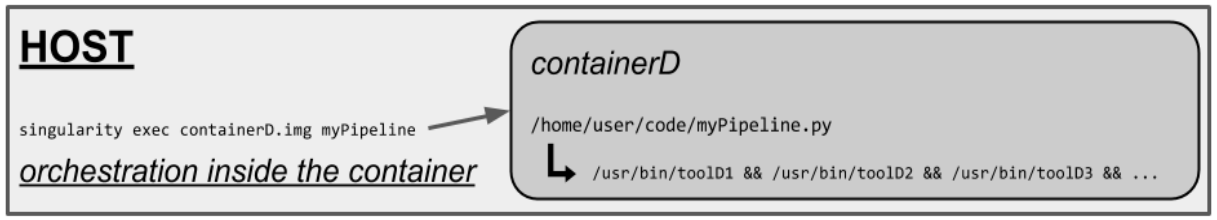


package tools inside the container and call them from your pipeline"

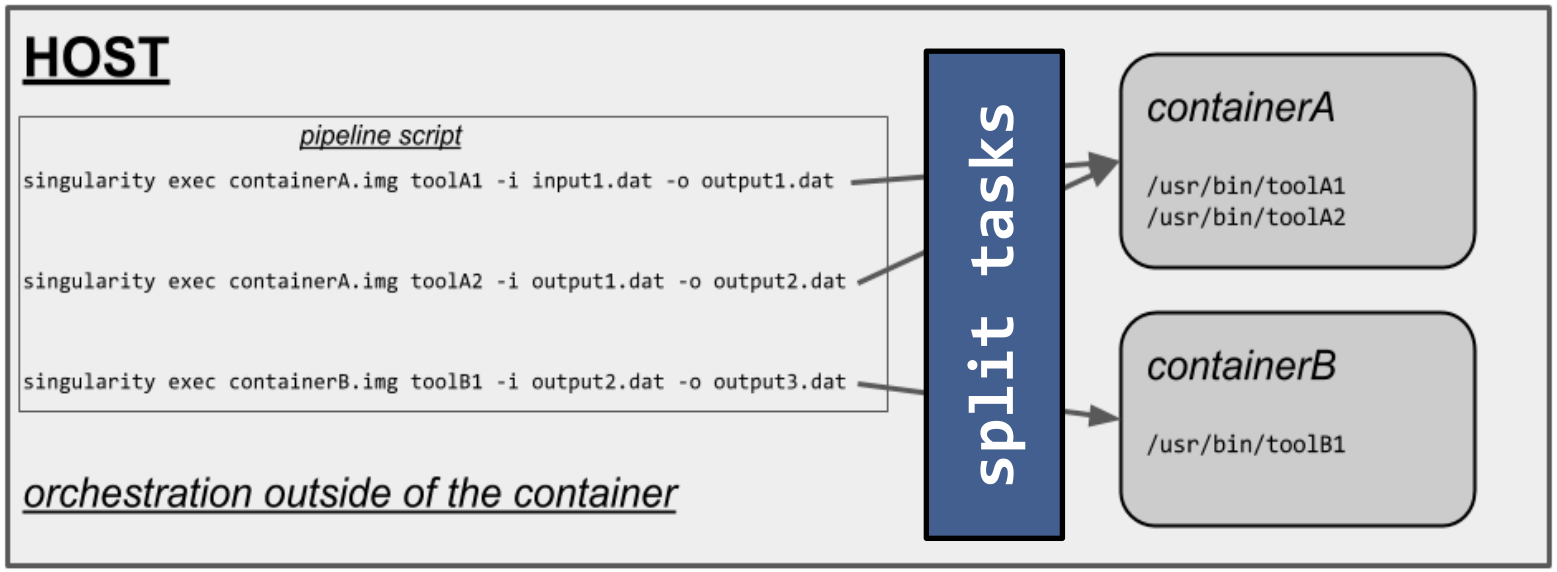
Pipeline Interoperability project – Orchestration strategies



package the whole pipeline inside the container and "just run it"



package the whole pipeline inside the container and execute it in a more flexible way (execute only parts of it, change options, etc.)



package tools inside the container and call them from your pipeline"

Outline

- Motivation
- Pipeline Interoperability

  e-Science Coordination Team

  pilot project

- EnhanceR project

  follow up to the pilot project

Pipeline Interoperability – follow up

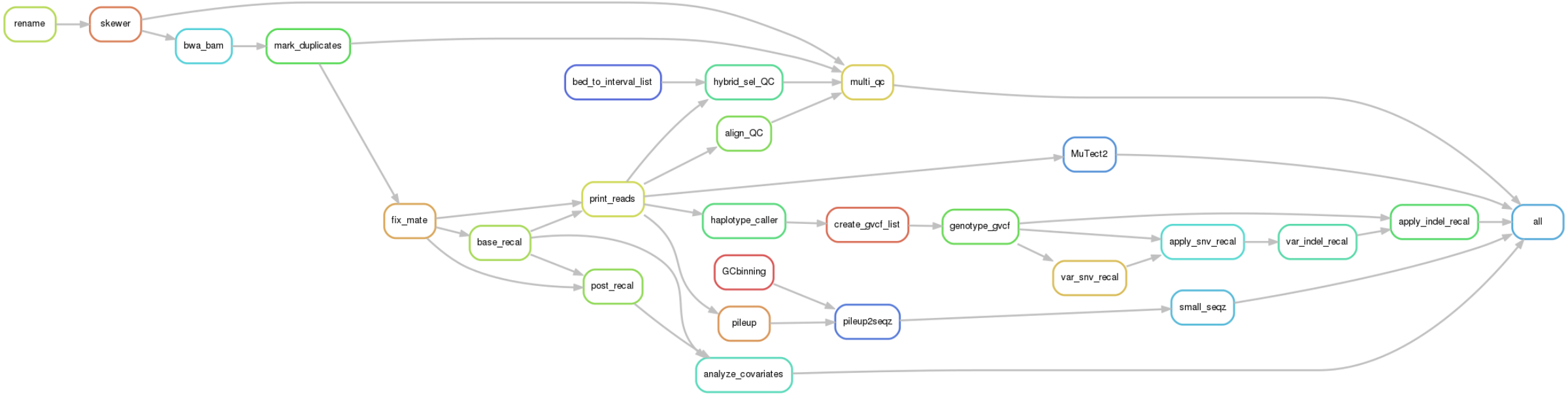
- *Coordination and Community*
- *MelArray*
  - a bioinformatics use case
- **GPU** Singularity containers
- **MPI** Singularity containers

MelArray – a containerized bioinformatics pipeline

EnhancER

UniversitätsSpital Zürich

- **Aim:** Convert a bash script based bioinformatics pipeline (*MelArray*) into a **containerized** Snakemake pipeline



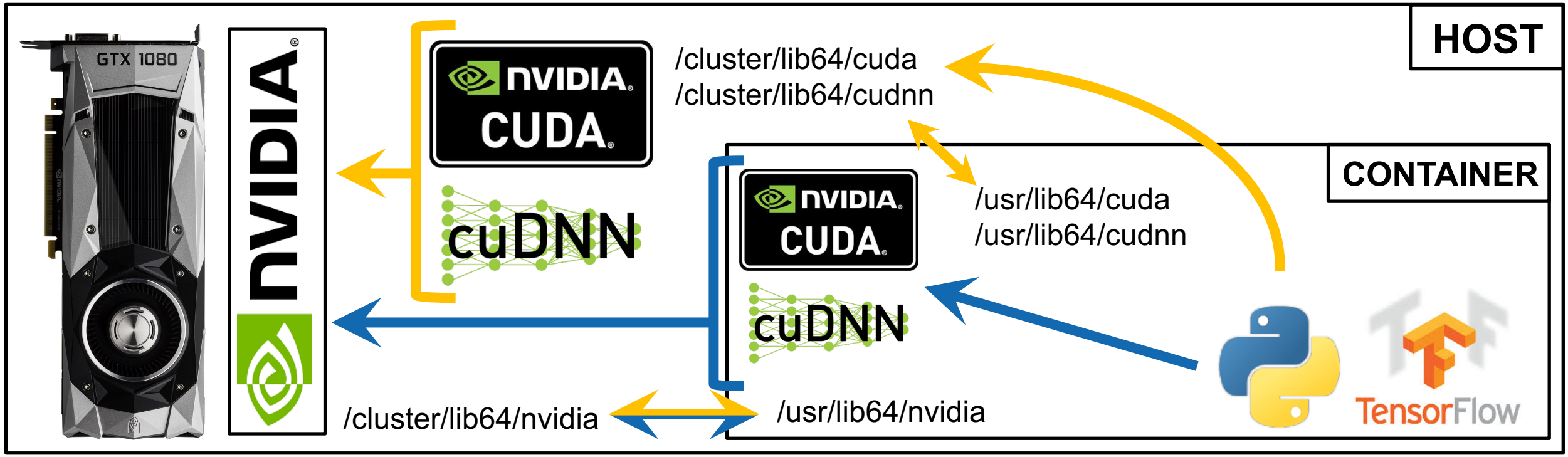
Pipeline Interoperability – GPU Singularity containers

- **GPU** Singularity container
  - ✅ Tested with TensorFlow and Python3
  - ✅ Tested with different versions of Python3, TensorFlow, CUDA
  - ✅ Testing done on the Leonhard cluster (on GTX 1080)

- **MPI** Singularity container
  -  Currently testing with openmpi 2.1.0

Pipeline Interoperability – GPU Singularity containers

- **GPU** Singularity container
  - ✔ Tested with TensorFlow and Python3
  - ✔ Tested with different versions of Python3, TensorFlow, CUDA
  - ✔ Testing done on the Leonhard cluster (on GTX 1080)



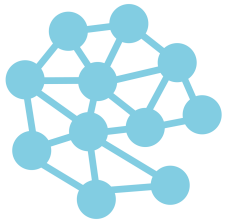
Conclusion

- Combining Docker and Singularity to achieve pipeline interoperability
  - technology scouting, best practices, guidelines / documentation
- Pipeline interoperability project is show case example of a successful collaborative effort within eSCT / EnhanceR
- Proof of concept that containers are a viable approach for the upcoming SPHN-related IT challenges

Contact

ETH Zürich
Balazs Laurenczy
Scientific IT Services
Weinbergstrasse 11
8092 Zürich

Guidelines document:
http://bit.ly/guidelines_containers



EnhanceR
www.enhancer.ch

Acknowledgements

eSCT / EnhanceR colleagues

Alexander Kashev (UniBe)
Pablo Escobar López (UniBas)
Sergio Maffioletti (UZH)
Heinz Stockinger (SIB)

SIS colleagues

Thomas Wüst
Samuel Fux
Urban Borstnik
Christiane Pousa