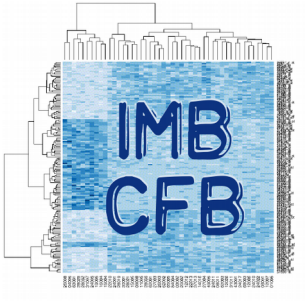
Keep moving forward

Sergi Sayols
Bioinformatics Core Facility
Institute of Molecular Biology Mainz

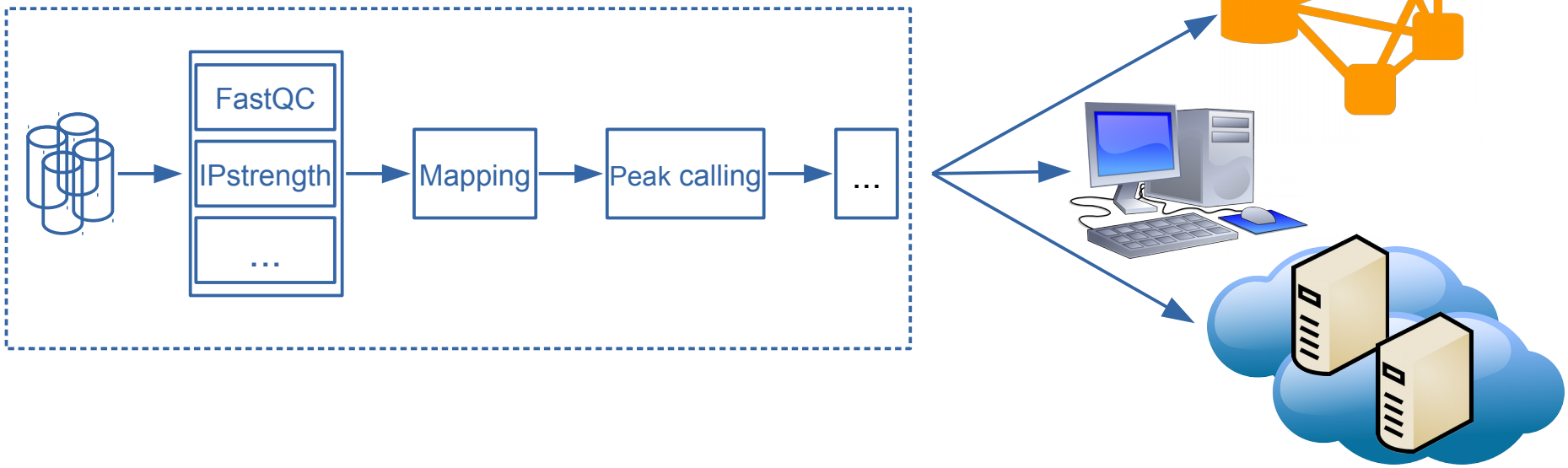
July 2016



Context

Our starting point:

- pipelines for standard tasks:
  - not all ChIP-seq experiments are the same, but share a common part: QC + Mapping + peak calling + …
  - same applies to other experiments: RNA-seq, Exome-seq, etc.
  - many options: bash, make, bpipe, snakemake, Galaxy, HTCondor
- infrastructure: workstations, servers, clusters, cloud



Context

Nice idea, let's think further

Challenges:

- making analysis reproducible over time
- changing versions
- new standards: Tophat + Cufflinks vs. Star + edgeR, Bowtie vs. BWA, …
- being able to reuse pipelines: new projects, new infrastructure
- scalability: +samples → +cores

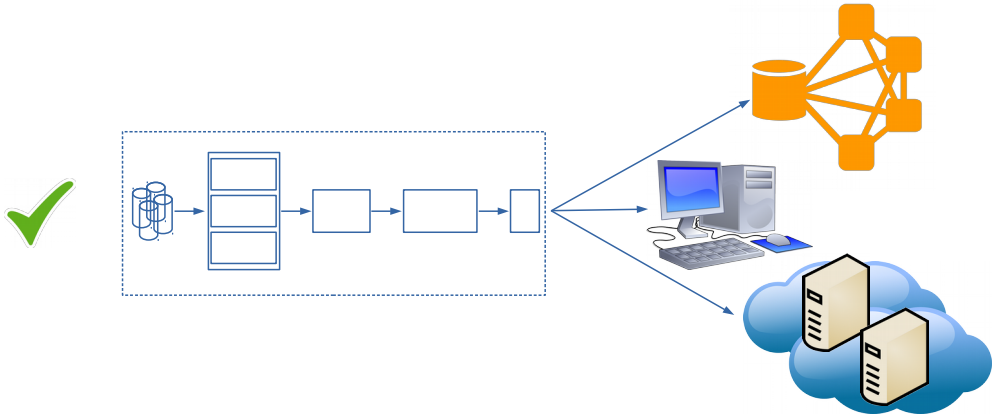
[one possible] Solution:

containerizing software stacks

Context

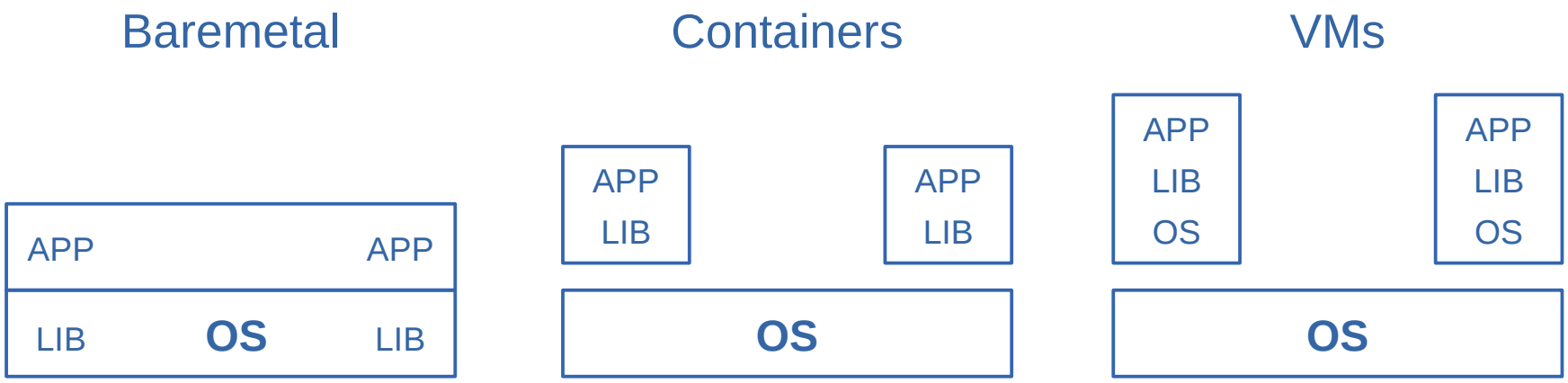
What's this talk about?

How to painlessly run all this
(painless to run, painful to set up)



- Analyzing a new project, as easy as copy & paste
- Getting the same results

Containers



- Tightly coupled
- Bare metal performance
- Not portable

- Loosely coupled
- Bare metal performance
- Portable

- Loosely coupled
- Performance overhead
- Not as portable

"Containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in."

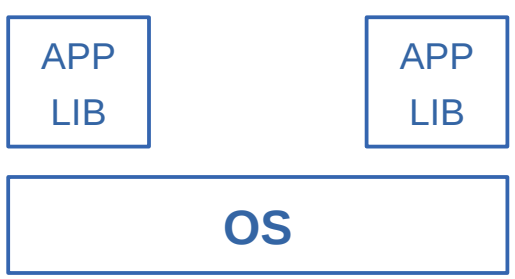
source: www.wikipedia.org

Containers

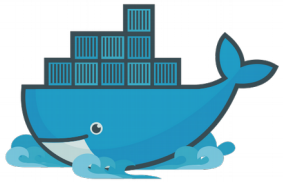
Institute of Molecular Biology

Benefits:

- ✔ lightweight VM
- ✔ image portability → pack once, run
- ✔ everywhere: workstation, server, cloud
- ✔ software stack exportable to tar.gz: pack with data
- ✔ control versions/dependencies of packaged software
- ✔ collaborate/branch/merge



Docker specific:



- ✔ build facility
- ✔ App containers: PaaS oriented single process container
- ✔ storage separated via layers

Containers

Institute of Molecular Biology

Downsides:

✗ security concern: need special group permissions. Possible solutions:

- grant permissions: works on some configs. OK
- up at boot time, add a listener. Complex, not cluster friendly

✗ complex integration with the batch scheduler. Partial solutions:

- run the full stack as a single task, with max resources allocated.
  Cons: no fine grained management of the resources
- containerize tools (samtools, bwa, R), not workflows/pipelines/stacks.
  Cons: more complex bundle/share/deploy

2 paradigms:

- pack single apps → friendly with LSF

✔ pack full software stacks → friendly with the data and the concept of PaaS

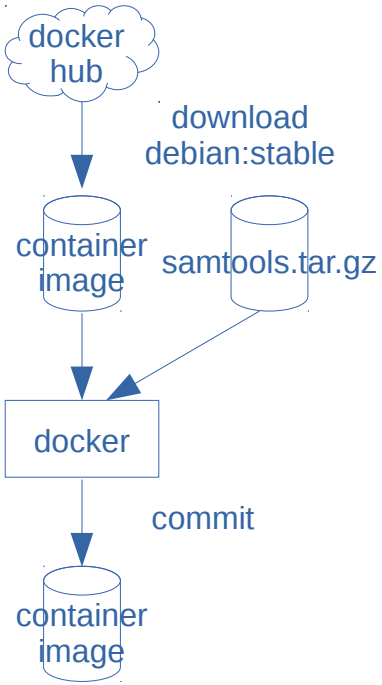
Working with docker containers

- Create a docker container: Dockerfile

```
FROM debian:stable
ENTRYPOINT ["/opt/bpipe/default/bin/bpipe","run"]

## install samtools
COPY ./deps/samtools-1.2 /usr/local/src/samtools-1.2
WORKDIR /usr/local/src/samtools-1.2
RUN     make && \
        make install prefix=/opt/samtools/1.2 && \
        rm -rf /usr/local/src/samtools-1.2 && \
        echo "export PATH=/opt/samtools/1.2/bin:\$PATH" > \
             /opt/samtools/1.2/env.sh && \
        chmod ugo+rx /opt/samtools/1.2/env.sh

## install pipelines and wrappers
COPY ./deps/imb-forge /opt/imb-forge
```



- Build:

```
$ docker build -t imbforge/chipseq:v1 .
```

- Push to the docker hub:

```
$ docker push imbforge/chipseq:v1
```

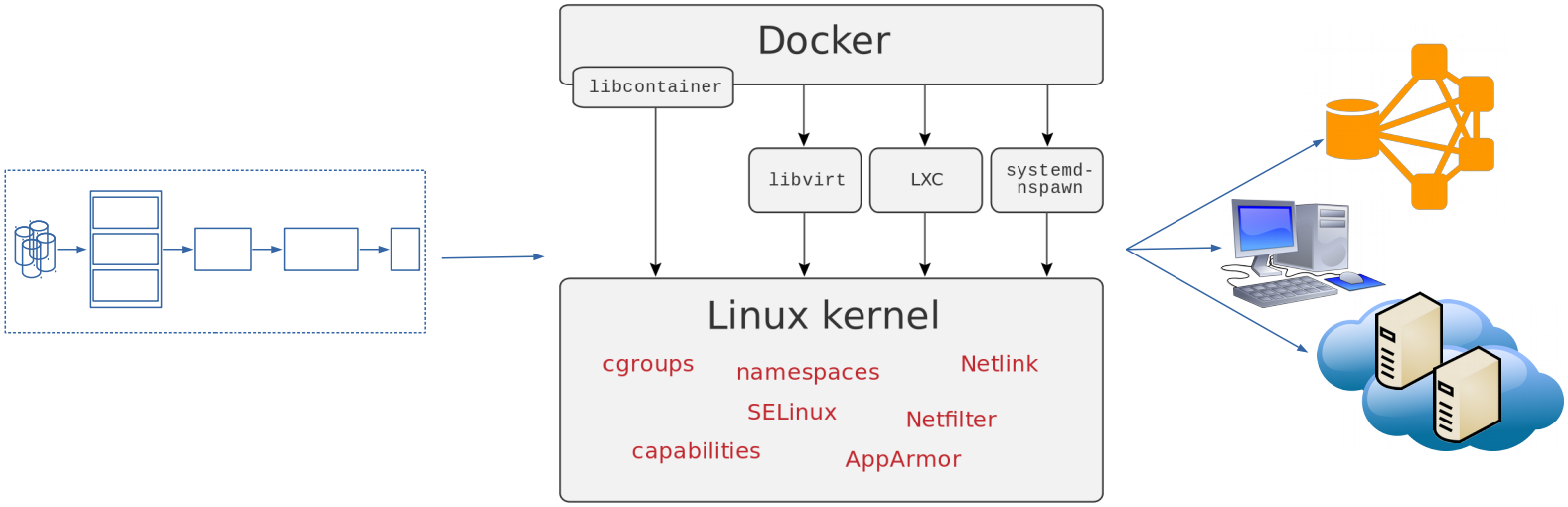
- Run a container:

```
$ docker run --rm -v ${WORKDIR}:${WORKDIR} -w ${WORKDIR} -t imbforge/chipseq:v1\
  -n ${MAX_PAR_PROCS} ${WORKDIR}/chipseq_v1.2.txt ${WORKDIR}/rawdata/*.fastq.gz
```

- Run an interactive shell:

```
$ docker run --entrypoint=/bin/bash -ti imbforge/chipseq:v1 -s
```

Working with docker containers



Containers in a workstation

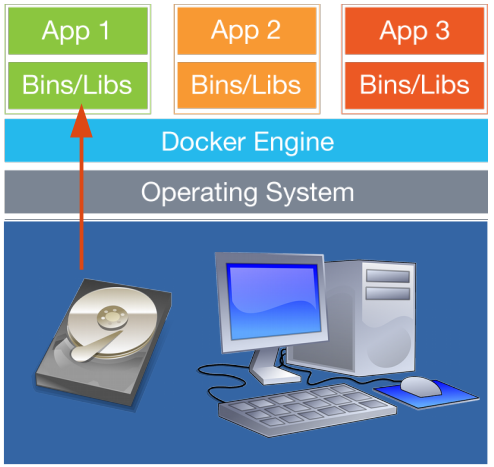
Pull & play:

- pull the container image from the repo

```
$ docker pull imbforge/rnaseq
```

- run with bind mount the local volume

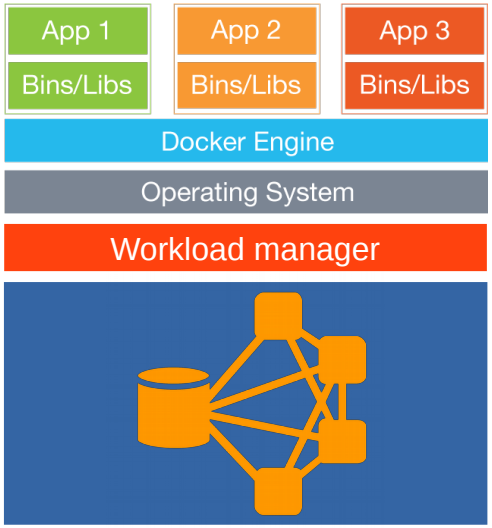
```
$ export WORKDIR=/project/rna-seq
$ docker run --rm -v ${WORKDIR}:${WORKDIR} -w \
  ${WORKDIR} -t imbforge/chipseq:v1 -n ${MAX_PROCS} \
  ${WORKDIR}/chipseq_v1.2.txt ${WORKDIR}/rawdata/*.fq.gz
```



Containers in a private cluster

Complex setup, many questions open:

- ✔ SAN/NAS/parallel FS help
- ✗ security concern: grant permissions or boot up the container with a listener
- ✗ difficult to fine-grain resource allocation
- possible solution (unexplored): the pipeline splits in batches and dynamically boots up containers
  - ✗ complex setup of the pipeline
  - ✔ suitable for large scale projects



Containers in a (semi)public cluster

- 555 nodes, 35,520 cores, 89TB RAM, 1110TB storage

✗ even worse, strict security policy!

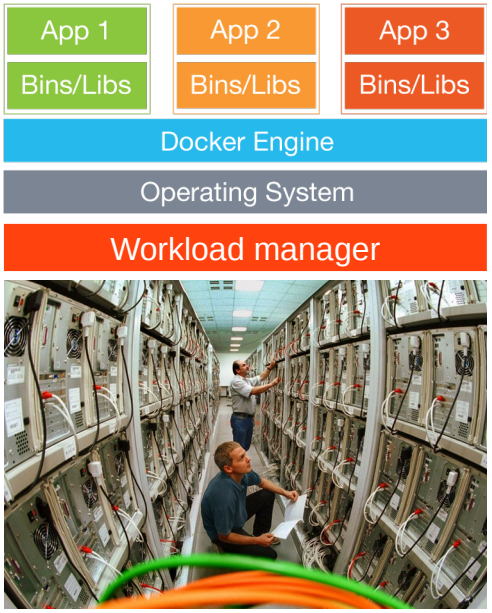
✗ no root-like for us

- Partial solution: chroot jails + special queues for the pipelines that automatically decompress a tarball with the software stack. Downsides:

  ✗ no fine-grained resource allocation

  ✗ CPU intensive (.tar.gz ~2Gb)

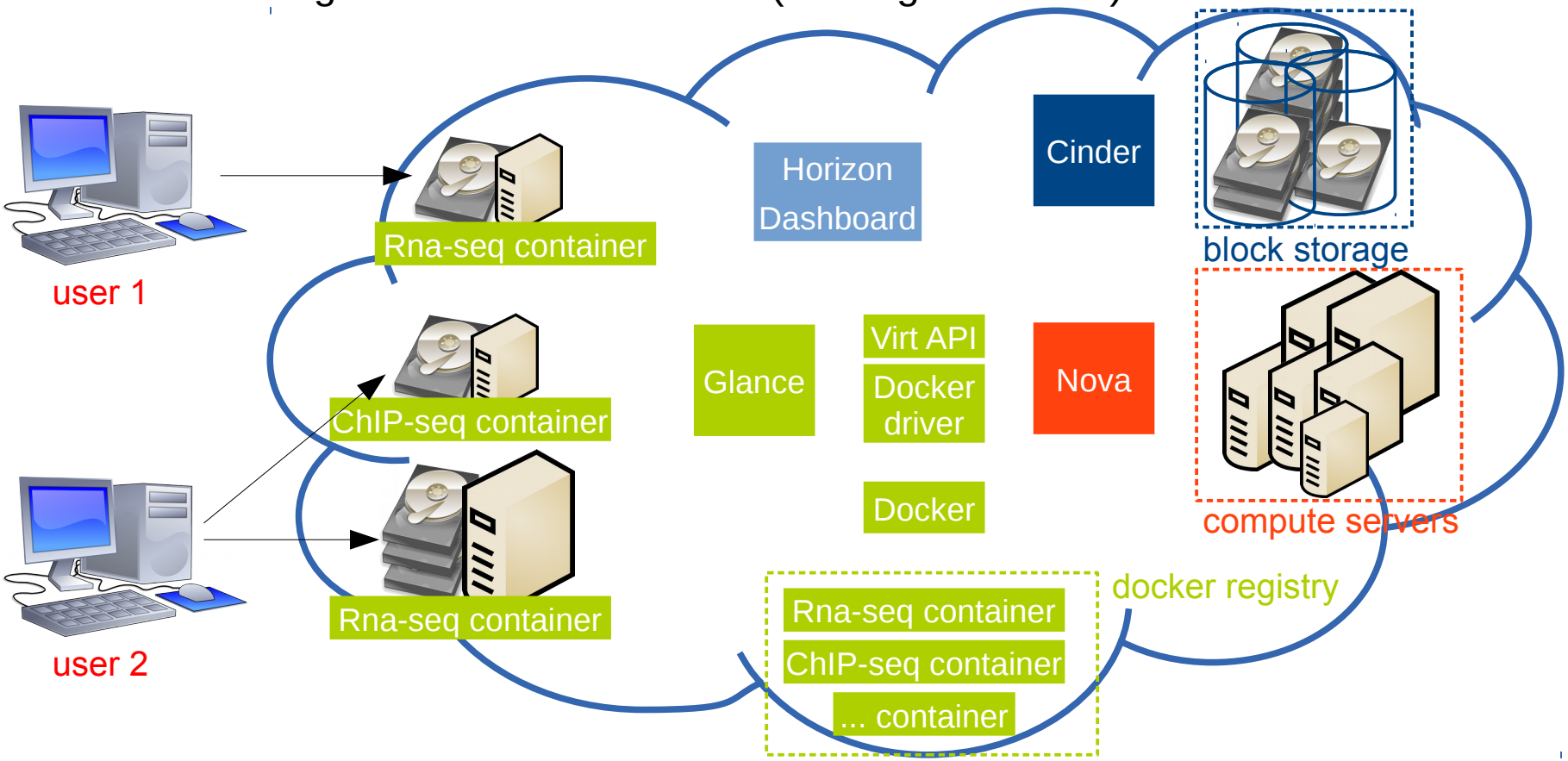
  ✗ complex setup



source: www.wikipedia.org

Horizon: Containers in the cloud

- PaaS: allow customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure
- load and boot the software stacks as containers
- easy/transparent to scale-up/down hardware, AWS
- better management of the resources (although not ideal)



Conclusions

- ✔ Containers are a neat way to deploy full application stacks
- ✔ Can be packed with the data to ensure reproducibility
- ✗ Not trivial to attach them to a batch scheduler
- ✔ The cloud paradigm may help to leverage the use of resources

Thanks!