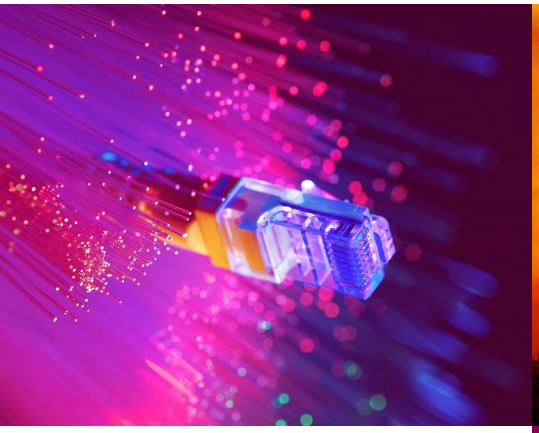
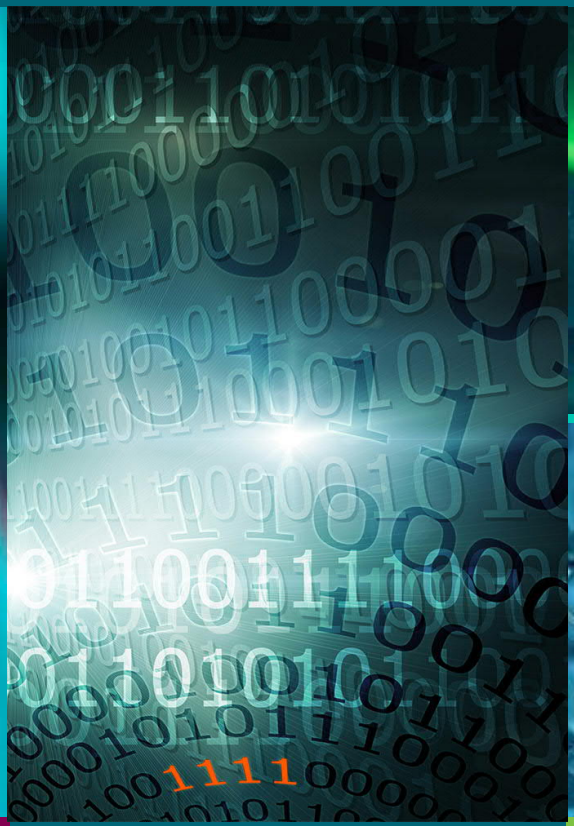
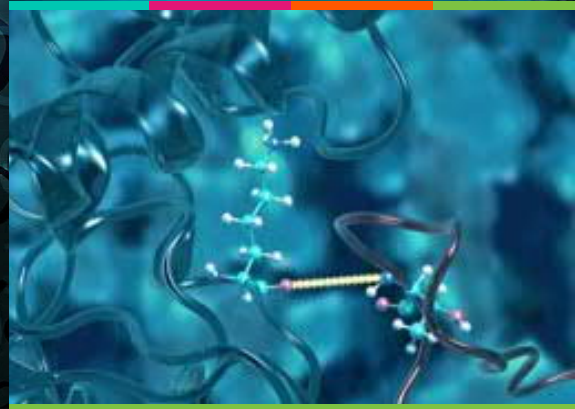


# Singularity

CSC – Suomalainen tutkimuksen, koulutuksen, kulttuurin ja julkishallinnon ICT-osaamiskeskus

# Singularity in a nutshell





- Designed for HPC environments
- Containers can be run with user level rights
  - Building new containers requires root access
- Minimal performance overhead
- Supports MPI
- Can use host driver stack (Nvidia/cuda)
  - Add option `--nv`
- Can import and run Docker containers

# Running Singularity containers

# Basic syntax

```
singularity run [run options...] <container>
```

Runs a script specified when container was built

```
singularity exec [exec options...] <container> <command>
```

Executes a command in the container

```
singularity shell [shell options...] <container>
```

Opens a shell in the container

# File system

- Containers have their own, internal file system
- To use host file system for input/output, host directories need to mapped to container directories

  `--bind /scratch/project_12345:/data`

  host directory `/scratch/project_12345` is mapped to

  directory `/data` inside the container
- Target directory inside the container does not need to exist. It is created as necessary
- More than one directory can be mapped if necessary

- Mounting container directories with the same path as host directories allows you to use same command line as you would without a container – but can be confusing when troubleshooting

```
singularity exec --bind /scratch/project_12345/data:/scratch/project_12345/data myimage.sif \
myprog --input /scratch/project_12345/data/myfile

singularity exec --bind $PWD:$PWD myimage.sif myprog --input myfile
```

- Using different name space for the container may be clearer, but you need to remember to use it in commands

```
singularity exec --bind /scratch/project_12345/data:/container/data myimage.sif \

myprog --input /container/data/myfile
```

Matter of taste: take you pick

# Using Docker containers with Singularity

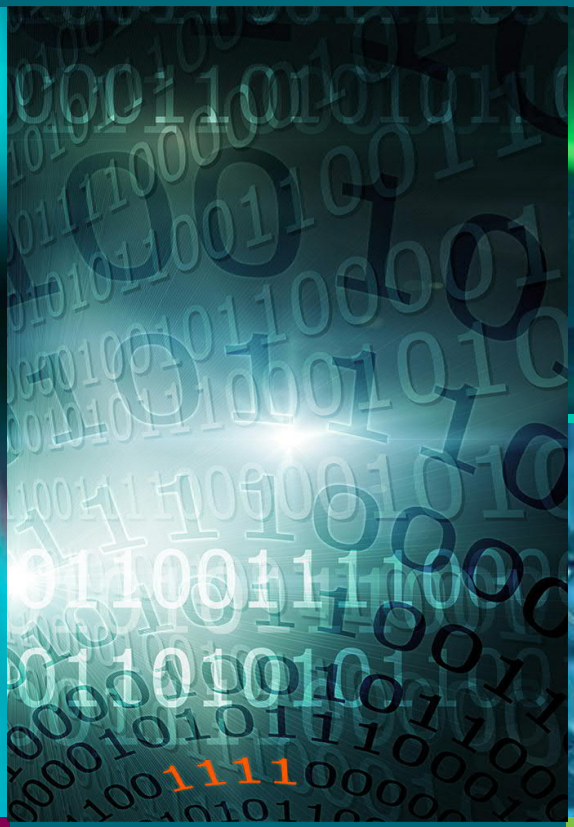
- You can build a Singularity container from a Docker container with normal user rights:

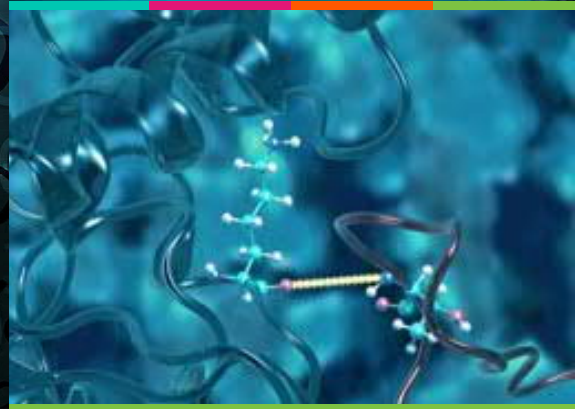
```
singularity build <image> docker://<address>
```

For example:

```
singularity build pytorch_19.10-py3.sif \
docker://nvcr.io/nvidia/pytorch:19.10-py3
```

# Comparison of installation methods





# Singularity containers as installation method

- Singularity is a good option in cases where installation is otherwise problematic:
  - Complex installations with many dependencies
  - Obsolete dependencies incompatible with general environment
    - Still needs to be kernel compatible
- Should be considered even when other methods exist

# Just a random example (FASTX-toolkit)

| Installation type | Size on disc | Number of files |
|---|---|---|
| Native | 1,9 MB | 47 |
| Conda | 1,1 GB | 27464 |
| Singularity | 339 MB | 1 |

- Containers are not the solution for everything, but they do have their uses…

# Building a new Singularity container

Requires root access:
Can not be done directly
In e.g. Puhti

- Typical steps

1. Build a basic container in sandbox mode ( `--sandbox`)
    1. Uses a folder structure instead of an image file
    2. Requires root access!
2. Open a shell in the container and install software
    1. Depending on base image system, package managers can be used to install libraries etc (`apt install`, `yum install` etc)
    2. Installation as per software developer instructions
3. Build a production image from the sandbox
4. (optional) Make a definition file and build a production image from it