

# Singularity containers and SLING: a tutorial

---

Blaž Škrlj

Jožef Stefan Institute

# Purpose of this tutorial

1. Get to know one of the common pitfalls of research code
2. How to mitigate this issue via dockerization
3. What are Singularity Containers?
4. **Hands-on:** Local builds
5. Cluster execution
6. **Hands-on:** Scaling up with SLING

# Part 1

## Code replicability and Singularity

# The four levels of repositories one finds these days

1. Messy, undocumented code with *no dependency specifications*
2. Code with dependency specifications, *not versioned*
3. Versioned code, dependencies
4. **Code with a replicable environment**

## (2) What is Singularity?

1. It's an environment for **dockerization**. Specifically focused on **scientific** and **HPC** environments.

In other words:

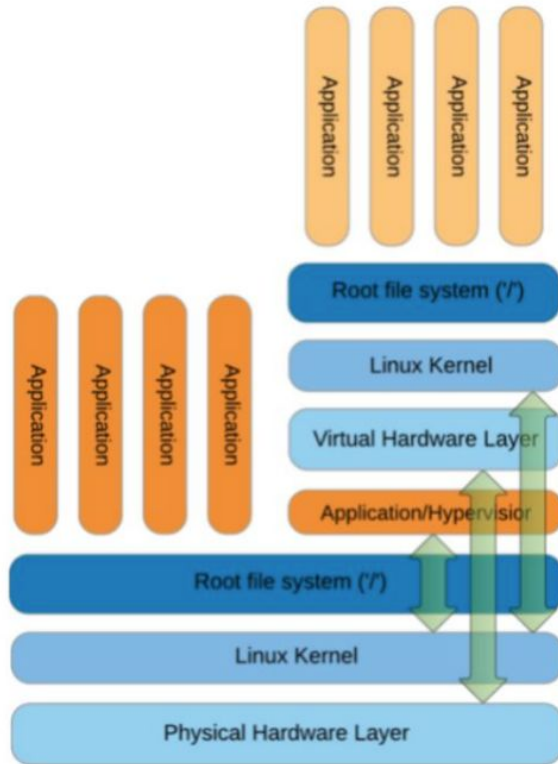
It's the engineering swiss knife suitable for the **fast-paced** (replicable) science.

# (1) What is Singularity?

1. Assuming you wish to share some code with a colleague
2. What's the minimal effort she/he needs to undergo to run the code?

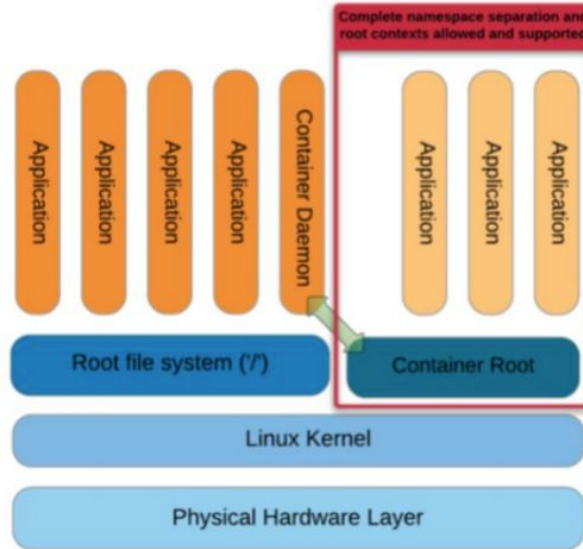


## Virtual Machine



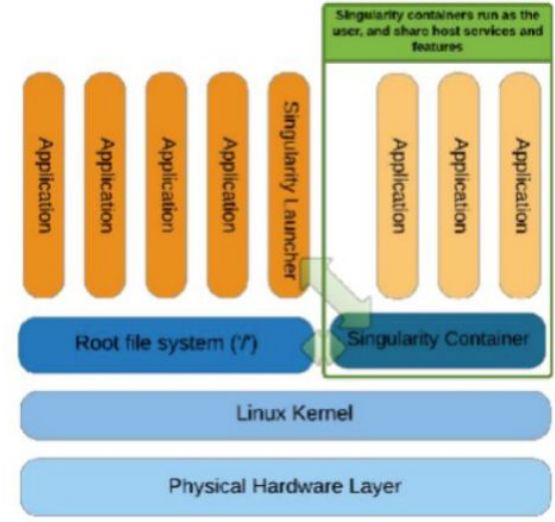
## Docker

- Independent namespaces
- Root supported contexts



## Singularity

- Host services and devices shared
- User access



## Summary

### When **to use** it

#### Testing

- Unstable releases
- Proof of concept executions
- Punctual data preproces

#### Reproducibility

- Complex software stacks

### When **NOT to** use it

#### Production

- Architecture aware software  
GROMACS, NAMD, CPMD...
- Frequent HPC applications

#### Not available soft?

- Request it to the support team

Detailed info at: <https://sylabs.io/docs/>



# Extremely useful for

Deployment on **multiple machines** (HPC).

Quick **replicable** demos and handling dependency problems.

“Oh, this machine is not doing anything, let’s just run this experiment” - type of adventures

# Tutorial time - part 1





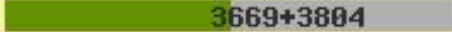



1. Installation
2. Simple generic replicability case
3. Building containers
4. Using GPUs

*Available at:* **<https://github.com/SkBlaz/singularity-container-tutorial#enter-singularity>**

# Part 2

## Singularity and SLING

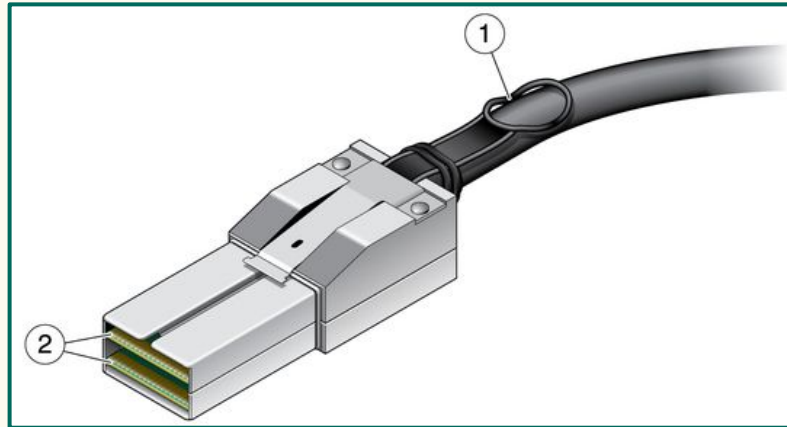
# SLING?

Site	CPUs	Load (processes: Grid+local)
ARNES	2048	 1845+8
CIPKeBiP	984	 144+8
Maister	9872	 200+8794
NSC	2880	 1424+548
SiGNET	8096	 3669+3884
SiGNET	8096	 3884+3669
Trdina	536	 188+16
UNG	240	 232+8
<b>8 sites</b>	<b>32752</b>	<b>11418 + 16831</b>

Learn more at: <http://www.sling.si/sling/en/eurocc/>

# Why Singularity in this environment?

1. Special support (InfiniBand)
2. No need for customized environments (SysAdmin support for each task)
3. Fast



Source: <https://docs.oracle.com/cd/E19654-01/835-0786-02/figures/127163.jpg>

# Key idea

1. Build the Singularity image locally, copy it to dCache
2. Each job sees the image and executes it (singularity exec [jobStuff])
3. The dCache version of the image can be updated by the user

If it works locally, it will **most likely** work on the grid too.

# Tutorial time - part 2

1. Let's build an image
2. Test it locally
3. Create XRS� job specification
4. Execute, retrieve and inspect the result

*Available at:*

**<https://github.com/SkBlaz/singularity-container-tutorial/blob/master/gpu.md>**

**<https://github.com/SkBlaz/singularity-container-tutorial/blob/master/optimization.md>**