Docker Vs Singularity

I’ve added more points about Singularity than Docker here.

Docker Pros:

- Very big community, easy troubleshooting, easier to find solutions to common problems
- Standardized, easier to find docker solutions for any usecase
- Easy to manage with Kubernetes
- Better for microservice architecture

Docker Cons:

- Root owned container daemon
- Have to mount the storage ourselves

Singularity pros:

- Good for running one container per virtual machine
- Single file based images(no layers) that contains everything
- Access and privilege: Same user inside container(not root inside container)
- Use host resources directly
- Automatically mounts /home directory
- Can access resources on the host
- Good for HPC context
- Can support mutiple user logging in into nodes
- Easy installation of dependencies (even Docker has)
- Similar environment (even Docker has)

· Easy portability, scalability (even Docker has)

· one img file that can be used and transported around easy to manage

· Better access control than Docker

· Good for multi-tenant computers

· Any user can run container without root privileges

· Integrates seamlessly into infrastructure (Ex. with Slurm, OpenMPI)

· Portability b/w many different systems (even Docker has)

· We can support users creating their own containers, using them

· Safe to run containers without screening their content as access is limited

· GPU access easy

· Multi-node MPI – have to check on this

· Can also import Docker containers.

Singularity solves the problem of users running their applications on HPC by introducing a layer of abstraction between the applications and the underlying operating system, while providing a safe passthrough to access computing resources required by ML, DL, and AI directly, with no performance penalty and without compromises on security.

Singularity cons:

· Not a very big community as Docker has

· Have to manage Singularity containers on our own as opposed to docker containers which are managed by Kubernetes

Why use containers?

To manage and run applications with complex dependencies easily and efficiently. It is easier to install software in containers. We can install libraries on the fly, different versions can be supported.

We can make use of Singularity containers in two ways:

1. We create an image for the environment with all the dependencies and start a container that is always running on each of the nodes. Then we use a batch scheduler like SLURM to execute and run parallel tasks inside the container. This ensures that our compute nodes all have the same software versions and packages installed.

2. The other case would be a model where we are running the containers as tasks under a HPC batch job queuing system such as SLURM. Our web application submits a task using the UI or the user submits a task by doing SSH into the cluster and the SLURM scheduler starts containers and executes task inside them.

Kubernetes and Docker would be advantageous if we are aiming for a microservice type of architecture, where we have several services running in different isolated containers (like geoserver, webapp) and interacting with each other using dedicated ports. However, if we are more inclined towards building a HPC kind of environment where there will be a compute nodes cluster which will be utilized by users to submit batch jobs(by them logging in into our VMs), and by one application(our web application) that will submit parallel jobs in the background, then we can make use of singularity as it has better user access control and can also utilize the GPU resources in an easier way than Docker.