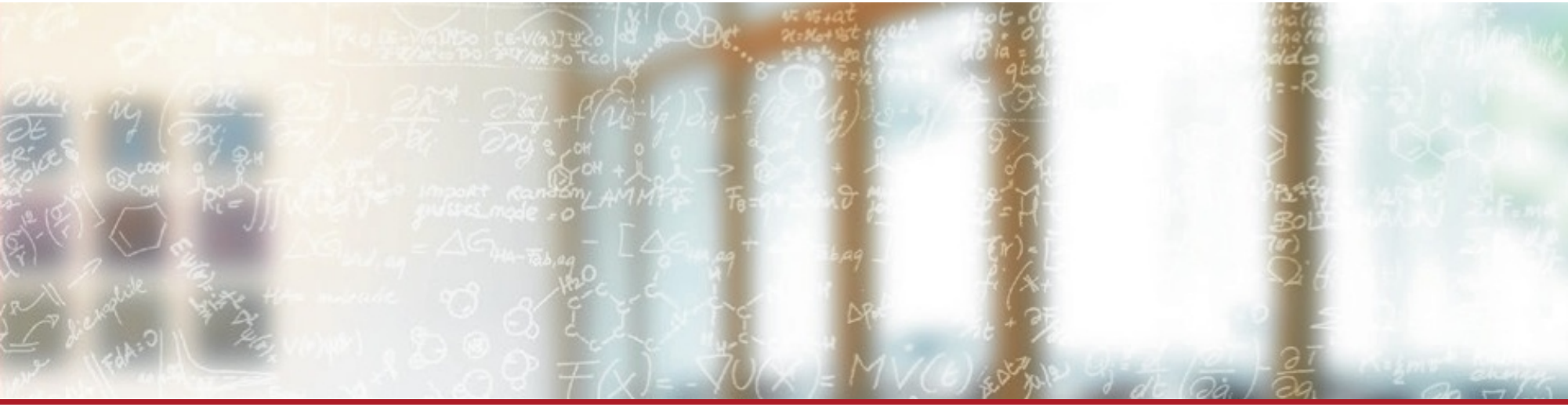
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



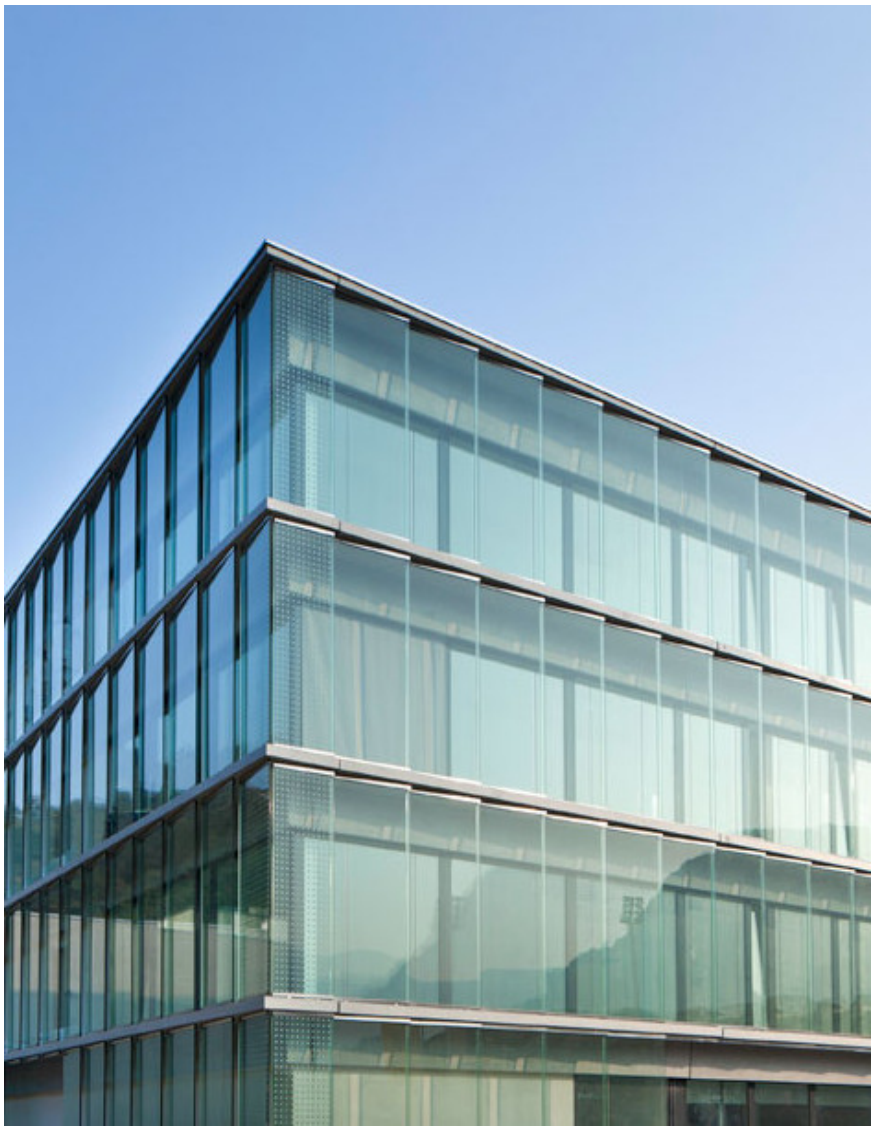
Shifter: Containers in HPC Environments

Second ADAC Workshop, Lugano, June 2016

Lucas Benedicic, CSCS

June 14th 2016

Agenda

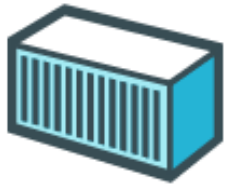


- Background
- Implementation
- Use Cases
- Security

Background

Docker overview

Build



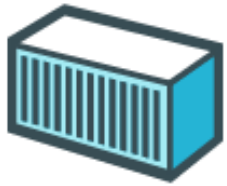
Ship

Run

- Build an image capturing all application requirements
- Commit the image or use a recipe file

Docker overview

Build



Ship

Run

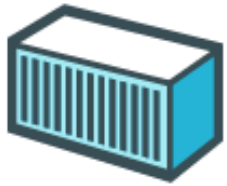
- Build an image capturing all application requirements
- Commit the image or use a recipe file

- Send the image descriptor to collaborators
- Push it to DockerHub or a private Registry

Docker overview

Build

- Build an image capturing all application requirements
- Commit the image or use a recipe file



Ship

- Send the image descriptor to collaborators
- Push it to DockerHub or a private Registry

Run

- Pull the image from DockerHub or a private Registry
- Launch the image as a container

Docker drawbacks

- **Architecture**
  Docker assumes
  a local disk

Docker drawbacks

- **Architecture**
  Docker assumes a local disk

- **Security**
  Docker users can easily escalate privileges on the host system

Docker drawbacks

- **Architecture**
  Docker assumes a local disk
- **Security**
  Docker users can easily escalate privileges on the host system
- **Integration**
  Docker is not designed to work with batch systems

Docker drawbacks

- **Architecture**
  Docker assumes a local disk

- **Security**
  Docker users can easily escalate privileges on the host system

- **Integration**
  Docker is not designed to work with batch systems

- **Complexity**
  Docker uses a client/daemon architecture

Solution: Shifter

- Partnership with NERSC and Cray to design a solution to run containers on HPC platforms

Solution: Shifter

- Partnership with NERSC and Cray to design a solution to run containers on HPC platforms
- Design goals
  - **Flexibility** requires no administrator assistance to launch a container

Solution: Shifter

- Partnership with NERSC and Cray to design a solution to run containers on HPC platforms
- Design goals
  - **Flexibility** requires no administrator assistance to launch a container
  - **Integration** shared resource availability (e.g., mounts, devices and network interfaces)

Solution: Shifter

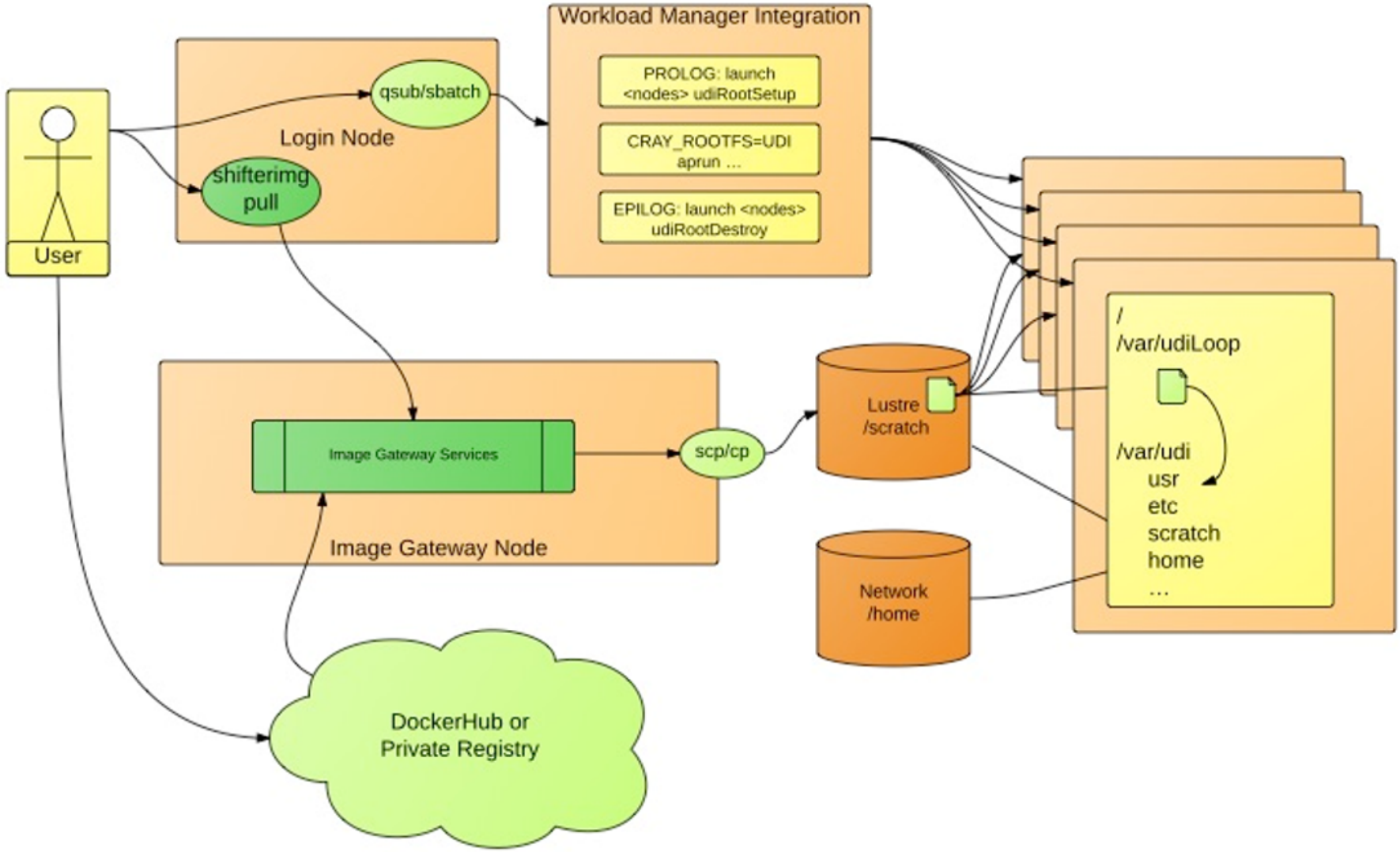
- Partnership with NERSC and Cray to design a solution to run containers on HPC platforms
- Design goals
  - **Flexibility** requires no administrator assistance to launch a container
  - **Integration** shared resource availability (e.g., mounts, devices and network interfaces)
  - **Compatibility** integrates with public image repositories (e.g., DockerHub)

Solution: Shifter

- Partnership with NERSC and Cray to design a solution to run containers on HPC platforms
- Design goals
  - **Flexibility** requires no administrator assistance to launch a container
  - **Integration** shared resource availability (e.g., mounts, devices and network interfaces)
  - **Compatibility** integrates with public image repositories (e.g., DockerHub)
  - **Security** stripped-down version of the original image is deployed in read-only mode

Implementation

Shifter Architecture



Shifter vs Docker: similarities

- The user-defined images are under user control

Shifter vs Docker: similarities

- The user-defined images are under user control
- Allows volume mapping
  - mount /a/b on the host on /b/a in the container

Shifter vs Docker: similarities

- The user-defined images are under user control
- Allows volume mapping
  - mount /a/b on the host on /b/a in the container
- Containers can be executed
  - environment variables, working directory, entry-point scripts, ...

Shifter vs Docker: similarities

- The user-defined images are under user control
- Allows volume mapping
  - mount /a/b on the host on /b/a in the container
- Containers can be executed
  - environment variables, working directory, entry-point scripts, ...
- Instantiate multiple containers on the same compute node

Shifter vs Docker: differences

- Containers run under the **user's UID** inside the container

Shifter vs Docker: differences

- Containers run under the **user's UID** inside the container
- Images are modified at construction time
  - Replaces `/etc/passwd, /etc/group, ...`
  - Generates `hostsfiles` to identify other nodes in the allocation

Shifter vs Docker: differences

- Containers run under the **user's UID** inside the container
- Images are modified at construction time
  - Replaces `/etc/passwd, /etc/group, ...`
  - Generates `hostsfiles` to identify other nodes in the allocation
- Images are read-only on the compute node

Shifter vs Docker: differences

- Containers run under the **user's UID** inside the container
- Images are modified at construction time
  - Replaces `/etc/passwd, /etc/group, ...`
  - Generates `hostsfiles` to identify other nodes in the allocation
- Images are read-only on the compute node
- Shifter does not use `cgroups` directly
  - Resources are handled by the workload manager (e.g., SLURM)

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Use Cases

Creating a Docker image

Dockerfile

```
FROM ubuntu:14.04
# Update packages and install dependencies
RUN apt-get update -y &&
apt-get install -y build-essential
# Copy in the application
ADD . /theapp
# Build it
RUN cd /theapp &&
make &&
make install
```

Use the image with Shifter

SLURM batch job

```
#!/bin/bash
#SBATCH -N 16 -t 20
#SBATCH --image=docker:ubuntu:14.04
#SBATCH --volume=/scratch/user/data:/data

module load shifter
srun -n 16 shifter /theapp/app
```

Shifter: Extending the Docker workflow to HPC

- Develop an application on your laptop and run it on a Supercomputer

Shifter: Extending the Docker workflow to HPC

- Develop an application on your laptop and run it on a Supercomputer
- Enables the user to define complex software-stacks themselves

Shifter: Extending the Docker workflow to HPC

- Develop an application on your laptop and run it on a Supercomputer
- Enables the user to define complex software-stacks themselves
- Runs the Linux flavor of their choice

Shifter: Extending the Docker workflow to HPC

- Develop an application on your laptop and run it on a Supercomputer
- Enables the user to define complex software-stacks themselves
- Runs the Linux flavor of their choice
- Improves reproducibility

Shifter: Extending the Docker workflow to HPC

- Develop an application on your laptop and run it on a Supercomputer
- Enables the user to define complex software-stacks themselves
- Runs the Linux flavor of their choice
- Improves reproducibility
- Improves sharing (e.g., Dockerfile, DockerHub)

Atlas and LHC

- CSCS operates a cluster running experiments of the LHC at CERN

Atlas and LHC

- CSCS operates a cluster running experiments of the LHC at CERN
- Jobs expect a RHEL-compatible OS and a precompiled software stack

Atlas and LHC

- CSCS operates a cluster running experiments of the LHC at CERN
- Jobs expect a RHEL-compatible OS and a precompiled software stack
- Shifter reproduces the complete software stack on the Cray XC

Atlas and LHC

- CSCS operates a cluster running experiments of the LHC at CERN
- Jobs expect a RHEL-compatible OS and a precompiled software stack
- Shifter reproduces the complete software stack on the Cray XC
- Job efficiency is comparable on both systems

Apache Spark

- Designed around commodity clusters, i.e., ethernet and local disks

Apache Spark

- Designed around commodity clusters, i.e., ethernet and local disks
- Does not scale well on parallel filesystems, e.g., Lustre

Apache Spark

- Designed around commodity clusters, i.e., ethernet and local disks
- Does not scale well on parallel filesystems, e.g., Lustre
- Shifter minimizes the metadata overhead

Apache Spark

- Designed around commodity clusters, i.e., ethernet and local disks
- Does not scale well on parallel filesystems, e.g., Lustre
- Shifter minimizes the metadata overhead
- Tested on NERSC's Cori up to 1600 nodes

Accessing GPUs

- Containers are both hardware-agnostic and platform-agnostic by design

Accessing GPUs

- Containers are both hardware-agnostic and platform-agnostic by design
- This is not the case when using GPUs
  - it uses specialized hardware, and
  - it requires specific software on the host, i.e., NVIDIA kernel driver

Accessing GPUs

- Containers are both hardware-agnostic and platform-agnostic by design
- This is not the case when using GPUs
  - it uses specialized hardware, and
  - it requires specific software on the host, i.e., NVIDIA kernel driver
- Shifter approach (CSCS + NVIDIA)
  - direct access to device characters
  - the required libraries are dynamically discovered at runtime

Accessing GPUs

- The Stream benchmark within a Shifter container shows native performance!
- NVIDIA's DGX-1 software stack is based on this solution

GPU: Stream benchmark

```
lucasbe@santis01 ~/shifter-gpu> sbatch ./nvidia-docker/samples/cuda-stream/benchmark.sbatch
Submitted batch job 496

lucasbe@santis01 /scratch/santis/lucasbe/jobs> cat shifter-gpu.out.log
Launching GPU stream benchmark on nid00012 ...
STREAM Benchmark implementation in CUDA
Array size (double precision) = 1073.74 MB
using 192 threads per block, 699051 blocks
Function      Rate (GB/s)  Avg time(s)  Min time(s)  Max time(s)
Copy:         184.3169     0.01167758   0.01165104   0.01170397
Scale:        183.1849     0.01175387   0.01172304   0.01178598
Add:          180.3075     0.01790012   0.01786518   0.01792288
Triad:        180.1056     0.01790700   0.01788521   0.01794291
```

Shifter and MPI

- Challenges
  - different versions, implementations (vendors), hardware ...

Shifter and MPI

- Challenges
  - different versions, implementations (vendors), hardware ...

- Embedded in the Image
  - add the required libraries into the image
  - users should maintain their own images

Shifter and MPI

- Challenges
  - different versions, implementations (vendors), hardware ...

- Embedded in the Image
  - add the required libraries into the image
  - users should maintain their own images
- Site-specific base images
  - users extend a managed image including the required libraries
  - these are upgraded together with the system

Shifter and MPI

- Challenges
  - different versions, implementations (vendors), hardware ...

- Embedded in the Image
  - add the required libraries into the image
  - users should maintain their own images
- Site-specific base images
  - users extend a managed image including the required libraries
  - these are upgraded together with the system
- Dynamic-linking at runtime
  - user's application built with ABI compatibility
  - system-specific implementation dynamically mounted at runtime

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Security

Container Security

- Security contexts don’t provide enough security and are difficult to configure, e.g., SELinux

Container Security

- Security contexts don’t provide enough security and are difficult to configure, e.g., SELinux
- Docker’s approach is broken by design, e.g., root in the container is still root in the host

Container Security

- Security contexts don’t provide enough security and are difficult to configure, e.g., SELinux
- Docker’s approach is broken by design, e.g., root in the container is still root in the host
- Look at what RedHat did

Shifter Security Model

- User accesses the container as their UID, not root or contextual root

Shifter Security Model

- User accesses the container as their UID, not root or contextual root
- Generated site /etc/passwd, /etc/group inside the container

Shifter Security Model

- User accesses the container as their UID, not root or contextual root
- Generated site /etc/passwd, /etc/group inside the container
- Embeded sshd is statically linked and accessible under the user's UID

Shifter Security Model

- User accesses the container as their UID, not root or contextual root
- Generated site /etc/passwd, /etc/group inside the container
- Embeded sshd is statically linked and accessible under the user's UID
- User-provided data are verified and filtered if needed, e.g., sudo

Conclusions

- Containers are here to stay

Conclusions

- Containers are here to stay
- Not an universal solution

Conclusions

- Containers are here to stay
- Not an universal solution
- They can be secured and isolated to match hypervisors

Conclusions

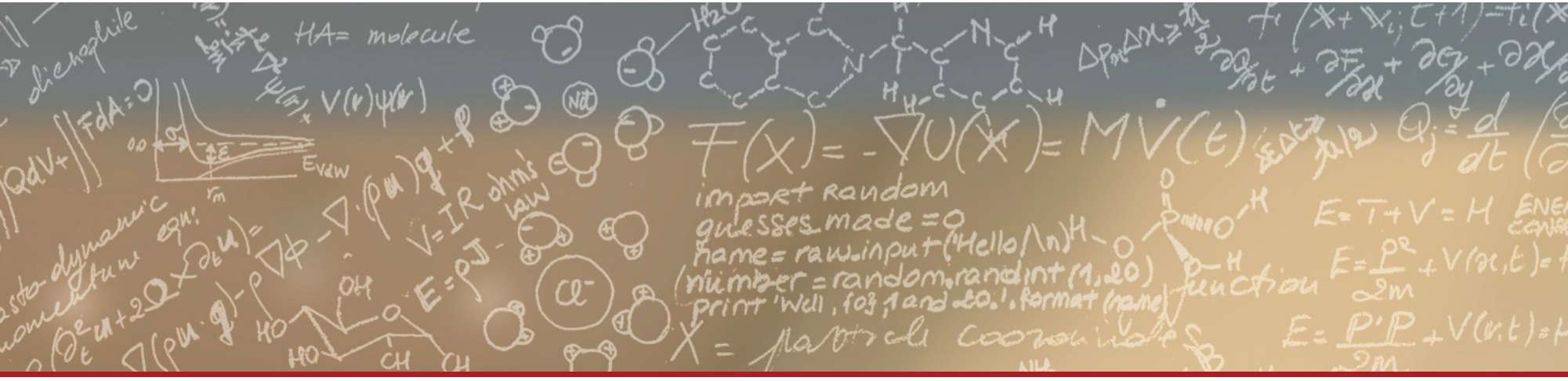
- Containers are here to stay
- Not an universal solution
- They can be secured and isolated to match hypervisors
- Still, several important issues arise

Conclusions

- Containers are here to stay
- Not an universal solution
- They can be secured and isolated to match hypervisors
- Still, several important issues arise
  - **Support**, e.g., how should images be maintained and/or troubleshooted?
  - **Adoption**, e.g., how are users adopting Docker?
  - **Training**, e.g., can we leverage from the Docker community? Is site-specific documentation needed?

Conclusions

- Containers are here to stay
- Not an universal solution
- They can be secured and isolated to match hypervisors
- Still, several important issues arise
  - **Support**, e.g., how should images be maintained and/or troubleshooted?
  - **Adoption**, e.g., how are users adopting Docker?
  - **Training**, e.g., can we leverage from the Docker community? Is site-specific documentation needed?
- **We haven't even scratched the surface of the possibilities**



Thank you for your attention.