Which tools and services do the HPAC Platform (and Fenix) offer?

1st HPAC Platform Training, 11-12 Dec 2018

Colin McMurtrie (CSCS)       Alberto Madonna (CSCS)
Overview of Services
Fenix/ICEI provides the Base Infrastructure for HPAC
What Services does Fenix/ICEI provide?

- **End-user Services**
  - Scalable Compute Services
  - Interactive Compute Services
  - SWIFT Object Storage
  - Data Storage Services
  - (Data Transfer Service) → HPAC
  - (Continuous Integration Services) → HPAC
  - (Software Packaging and Deployment Services) → HPAC
  - {Visualisation Services} → HPAC

- **Platform Services**
  - Infrastructure Services (middleware access to HPC resources via RestAPIs)
  - Infrastructure as a Service (e.g. OpenStack) for Virtual Machine Services
  - Data Management Services
  - User and Resource Management Services
  - Service Accounts (currently not available at all sites)
ICEI Resources for HBP

- ICEI resources have already been made available to the HBP (highlighted in green) and PRACE by CSCS
  - There are currently 8 HBP projects with compute allocations at CSCS
    - More are in the approval stages
  - More resources are available than are being consumed so HBP users are encouraged to apply for a compute allocation
    - More on this in the next session
How do I use ICEI Resources? (1)

**Scalable Compute Resources:**
The *Piz Daint* system is available as a state-of-the-art scalable compute resource for use by HBP users

- Accessible globally via Command-line Interface
- Via the Unicore GUI
- Via the RESTful API offered via UNICORE for platforms
  - Use of Service Accounts for Platforms is also acceptable at some sites (e.g. CSCS)
  - See next slide for some more details
How do I use ICEI Resources? (2)

**Interactive Compute Resources:**
The *Piz Daint* system supports the use of Jupyter Notebooks for interactive supercomputing, powered by JupyterHub

- This is a multi-user Hub that spawns, manages and proxies multiple instances of the single-user Jupyter notebook server
  - More details below
- Sessions later in the day will demonstrate the use of this environment

*Piz Daint* and other HPAC HPC systems are also accessible from the Jupyter Notebooks service of the *Collaboratory*

- This relies on the RESTful API offered via UNICORE for platforms
- The session later this morning will go into the details of how to do this
**Pollux OpenStack IaaS:**
The Pollux OpenStack IaaS is available to host your platform VMs:
- Accessible globally via the Horizon GUI interface
- RESTful API can be used for automation

Example of a Platform service (NRP) using VMs AND HPC resources.
How do I use ICEI Resources? (3)

**Swift Object Storage:**
SWIFT OS can be accessed directly from your personal computer
- GUI clients e.g. CyberDuck
- SP5 Python Library
  - Better for mgmt. of the ACLs and Object Buckets

Reachable from inside the **Collaboratory**
- Get/Put from Jupyter Notebooks
- More capabilities coming soon
How do I use ICEI Resources? (3)

Active Data Repositories:
• Come as part of the compute allocation (= $SCRATCH)
• Low-latency storage tier (Cray DataWarp with SSDs) in *Piz Daint* can also be requested

Archival Data Repositories:
• Are available either as part of a computing request (your proposal should state how much you need)
• Separately in a data-storage only use case (in which case a separate proposal is needed)
Service Detail: Software Packaging and Deployment
Containers

- Lightweight, isolated environments to run applications/services
- Already include all software dependencies
- Interest from HPC: a way to provide user-defined software stacks
Container implementations

- OpenVZ
- LXC
- Singularity
- rkt
- ORACLE Solaris
- Charliecloud
- SHIFTER

HPC focused
Docker

- Extremely popular container implementation

- Easy to use authoring tools
  - Container images are created from recipe-like files
  - Images can be named, tagged and built on top of other images

- Cloud-based image distribution strategy
  - Several remote registries available (e.g. Docker Hub)
  - Client includes facilities to authenticate, push and pull images
Docker workflow

1. An image is created locally from a *Dockerfile*

2. Push (i.e. upload) the image to a remote registry
   - DockerHub is the public registry maintained from the Docker company

3. Pull (i.e. download) the image on a target machine and run the container
Key terms

- **Image**: standalone, executable package that includes everything needed to run a piece of software
  - code, runtime libraries, environment variables, configuration files

- **Container**: runtime *instance* of an image
  - What the image becomes in memory when actually executed
  - Runs completely isolated from the host environment by default
    - only accessing host resources if configured to do so
So... how are containers useful?

Containers give the possibility to create (scientific) applications that are:

1. Portable
2. Reproducible
3. Easy to deploy
4. Easy to test

Unfortunately Docker containers are not a panacea for HPC environments because of:

- Security concerns
  - root in the container means root on shared parallel file systems
- Performance Portability
  - Performance is important in HPC (it’s in the name...)
Shifter

- Shifter is a *Docker-compatible* container platform specifically developed for HPC and addressing:
  - Security
  - Accounting
  - Native performance from custom HPC hardware
  - Integration with site infrastructure

- Enables flexible and convenient user workflows:

1. Create Docker image
2. Push to Docker Hub
3. Pull into storage at HPC center
4. Run at scale on HPC system
Shifter development @ CSCS

- The Infrastructure & Development Services group works on extending Shifter with a focus on:
  - Usability
  - Features
  - Performance

- Previous work:
  - Native GPU support: automatic import of host’s CUDA driver and devices
  - Native MPI support
    - Transparently swap container’s MPI libraries with the host’s at runtime
    - Enables full performance from vendor-specific implementations (e.g. Infiniband, Cray Aries)
Shifter development @ CSCS - (cont.)

- Software Architecture
  - Single executable, no background service
  - Image Manager component: robust, fast, designed from scratch

- Docker-like command line interface

- Improved container customization
  - User-specified mounts
  - “Writable volatile” directories
Shifter Architecture - CSCS branch

Docker Image registry

HPC System

Shifter Image Manager

Shifter (CLI)

Shifter Runtime

Parallel filesystem

Compute nodes

User

Workload Manager
Shifter Image Manager

- Container image management component written in C++
- Pull/query/remove images in user owned repositories
- Import images from tar files
- Parallel and robust layer download
  - automatic retry in case of errors
- Improved image expansion and local filesystem use
Image Manager performance

- **Image**: NVIDIA CUDA 8.0 Toolkit on CentOS 7 (official image)
- **Size on Docker Hub**: 1 GB (6 layers)
- **Total speedup**: 3.73x
**Image Manager performance**

- **Image:** Microsoft Cognitive Toolkit (CNTK) custom build
- **Size on Docker Hub:** 6 GB (32 layers)
- **Speedup:** 4.20x
Shifter CLI

- Command line processing component
- Goal was providing an in interface as close as possible to Docker
  - Consistent experience
  - Smoother transition between platforms
**CLI comparison**

### Shifter

# run container
$ shifter run [options] <image>[:tag]
<args>

# pull image
$ shifter pull [options] <image>[:tag]

# show list of images
$ shifter images

# remove image
$ shifter rmi <image>[:tag]

# import image
$ shifter import [options] <file> <image>

### Docker

# run container
$ docker run [options] <image>[:tag]
<args>

# pull image
$ docker pull [options] <image>[:tag]

# show list of images
$ docker images [options] [repo[:tag]]

# remove image
$ docker rmi [options] <image> [image...]

# import image
$ docker import [options] <file>|<URL>|...
Support for private & 3rd party registries

- Authentication option for private registries (--login)

```bash
$ shifter pull user/privateRepo:tag --login
username : user
password :
...
```

- Support for 3rd party registry services
  - $ shifter pull <server>/<namespace>/<image>:<tag>
  - e.g. NVIDIA GPU Cloud

```bash
$ shifter pull nvcr.io/nvidia/caffe:17.12 --login
username : $oauthtoken
password :
...
```
Shifter Import

- Import image from a tar file created by `docker save`
- Deploy an image to the HPC system without using the cloud

```
$ shifter import ./debian.tar my_debian

> expand image layers ...
> extracting :
/tmp/debian.tar/7e5c6402903b327fc62d1144f247c91c8e85c6f7b64903b8be289828285d502e/layer.tar
> make squashfs ...
> create metadata ...
# created: <user dir>/.shifter/images/import/library/my_debian/latest.squashfs
# created: <user dir>/.shifter/images/import/library/my_debian/latest.meta
```
Container customizations
User-specified Mounts

- Map some paths from the Host to another location within the container
- Requested at launch time with the `--mount` option
- Reproduces the same option syntax from Docker

```bash
$ ls -l /data
-rw-r--r--. 1 root root 1048576 Feb 7 10:49 data1.csv
-rw-r--r--. 1 root root 1048576 Feb 7 10:49 data2.csv

$ shifter run --mount=type=bind,source=/data,destination=/input debian bash

[user@container]$ ls -l /input
-rw-r--r--. 1 root 0 1048576 Feb 7 10:49 data1.csv
-rw-r--r--. 1 root 0 1048576 Feb 7 10:49 data2.csv
```
Writable volatile directories

- Directories originating from the container image are mounted as read-only

- Some use cases have specific requirements (e.g. create file in /var/lock)

- The --writable-volatile option of shifter run can be used to make such directories writable

- Original contents of the directory keep owners and permissions, but it is possible to create new files and work with them (thus, “writable”)

- Any modification made to the directory is lost when the container exits (thus, “volatile”)

Writable volatile directories

$ shifter run --writable-volatile=/usr/local debian bash

[user@container]$ ls -l /usr
-rw-r--r-- 1 <user name> <group name> 0 3560 Oct 9 00:00 bin/
-drwxr-xr-x 2 root 0 3 Jul 13 13:01 games/
-drwxr-xr-x 2 root 0 3 Jul 13 13:01 include/
-drwxr-xr-x 20 root 0 324 Oct 9 00:00 lib/
-drwx------ 10 <user name> <group name> 105 Oct 9 00:00 local/
-drwxr-xr-x 2 root 0 961 Oct 9 00:00 sbin/
-drwxr-xr-x 41 root 0 670 Oct 9 00:00 share/
-drwxr-xr-x 2 root 0 3 Jul 13 13:01 src/

[user@container]$ echo "Hello world" > /usr/local/hello.txt
[user@container]$ ls -l /usr/local/
...-rw-r--r-- 1 <user name> <group name> 12 Dec 19 15:18 hello.txt
...

[user@container]$ cat /usr/local/hello.txt
Hello world
Wrap-up

- **Improved deployment and operation**
  - Simpler architecture
  - Streamlined build/installation process
  - No background service

- **Improved user experience**
  - Docker-like CLI for a more consistent workflow
  - Robust, faster image pulling
  - Import images bypassing the cloud
  - Support private and 3rd party repositories
  - User owned image repositories improve privacy
  - Mount custom directories in the container
  -Writable volatile directories

- **More information available at**
  - https://user.cscs.ch/tools/containers/
Cheatsheet

Step-by-step guides: https://github.com/eth-cscs/containers-hands-on

docker pull <repo/image:tag>
docker run <image:tag> <command>
docker run -it <image:tag> bash
docker run <image:tag> mpiexec -n 2
docker images
docker build -t <repo/image:tag> .
docker login
docker push <repo/image:tag>
Shifter is not just for HPC!

```
pi@raspberry pi: ~ $ cat /etc/os-release
PRETTY_NAME="Raspbian GNU/Linux 9 (stretch)"
NAME="Raspbian GNU/Linux"
VERSION_ID="9"
VERSION="9 (stretch)"
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL="http://www.raspbian.org/RaspbianBugs"
pip@raspberry pi: ~ $ shifter images
REPOSITORY TAG DIGEST CREATED SIZE SERVER
arm32v6/alpine latest 7a643060ae76 2018-03-24T19:05:42 1.68MB index.docker.io
pi@raspberry pi: ~ $ shifter run arm32v6/alpine cat /etc/os-release
WARNING: skipping mount of image's /home. The file or directory already exists in the container's resources in the container whose path conflicts with the contents of the image.
WARNING: skipping mount of image's /tmp. The file or directory already exists in the container's resources in the container whose path conflicts with the contents of the image.
NAME="Alpine Linux"
ID=alpine
VERSION_ID=3.7.0
PRETTY_NAME="Alpine Linux v3.7"
HOME_URL="http://alpinelinux.org"
BUG_REPORT_URL="http://bugs.alpinelinux.org"
pip@raspberry pi: ~ $ _
```
Service Detail:
Continuous Integration
Jenkins CI Overview

- CSCS provides the Java-based open source Jenkins interface as an automation server
  - Can be used as a simple continuous integration (CI) server or turned into a CI tool for projects
  - Each project is assigned a Jenkins folder with the corresponding project name on the Jenkins instance
  - The Jenkins jobs related to the project have to be created in the above folder
  - Credentials can be added to be used with version control systems, etc.
  - Each project is assigned a Jenkins node which will manage the corresponding Jenkins jobs
  - Each project is additionally assigned a Jenkins user which is going to be used by the Jenkins node to access Piz Daint

- Since the CSCS Jenkins is not visible in public web, it is not possible to communicate with Github and trigger builds via webhooks. Two alternatives are recommended:
  - Use polling with a reasonable timestep to poll your remote repository for changes.
  - Use the GitHub Pull Request Builder (ghprb) plugin
Overview of Jenkins Service Interactions

User-perspective Interactions

Service account
Per project "jenkXX"

CSCS hpc systems
- Piz Daint
- Dom (TDS)

Jenkins VM

Jenkins webserver

Pull request
user GitHub account

GitHub

User

perspective Interactions

sbatch

ela.cscs.ch

ssh tunnel

ssh server

ssh

Jenkins

dashboard [Jenkins]
Service Detail:
JupyterHub Service at CSCS
Using JupyterHub at CSCS

- This service enables the interactive execution of Jupyter Notebook on *Piz Daint* over both single and multiple nodes.
  - The supported python version is python3.

- The service is accessed through the address
  - [https://jupyter.cscs.ch](https://jupyter.cscs.ch)
  - users should provide their HPAC credentials in order to login

- Once logged in, the user is redirected to a job setup page
  - Allows typical job configuration options to be selected in order to allocate the resources that are going to be used to run Jupyter
    - account
    - type of *Piz Daint* node type (gpu or mc)
    - number of nodes
    - wall-clock time limit

- More information at: [https://user.cscs.ch/tools/interactive/](https://user.cscs.ch/tools/interactive/)
JupyterHub Service Architecture (1)

- The **current** architecture protects the notebook in each compute node (CN) by launching a JupyterHub Service along with it.
Notebooks v4.3 and newer are protected with a per-session tokens

- Avoids the creation of several custom spawners
  - Ideally we want one CSCS spawner only
- Will be integrated with an Infrastructure Services API (UNICORE or similar)
- The frontend will be kept outside of the HPC system
How to get Help or More Information

General Contact for HPAC Platform:
• HPAC Platform:  
  https://collab.humanbrainproject.eu/#/collab/264/nav/2378

How to apply for resources:
• Send your proposals to: icei-coord@fz-juelich.de

Getting help:
• Send emails to: hpac-support@humanbrainproject.eu
Thank You

colin@cscs.ch       madonna@cscs.ch

www.humanbrainproject.eu       @HumanBrainProj       Human Brain Project