Enhancing Analysis Performance and Reproducibility

with Containers and the Cloud

Ricardo Rocha, CERN IT - ExaHealth 2021

https://indico.cern.ch/event/1078121

Ricardo Rocha

Computing Engineer - CERN Cloud Team

- Kubernetes and Containers, Networking and SDN
- GPUs and other Accelerators, Machine Learning

Cloud Native Computing Foundation (CNCF)

- Representative of CERN in the CNCF and End User Community
- Member of the CNCF Technical Oversight Committee (TOC)
- https://www.cncf.io/people/technical-oversight-committee/
- Lead of the CNCF Research User Group
- https://github.com/cncf/research-user-group

@ahcorporto ricardo.rocha@cern.ch

Will the infrastructure run my software in 10 years?

Machine	Default	Optimization off	Maximum	'Value' of -O
IBM/AIX	noopt		-O3	-O2
HP/UX	noopt		+O3	+O2
Solaris	noopt		-O4	-O3
Tru64 UNIX (Digital-UNIX)	-O4	-O0	-O4	-O4
SGI	-O1	-O0	-O3	-O2

7.3 Important Platform Dependent Differences

On most platforms at CERN the recommended Fortran compiler is called **f77**. The exception is HP/UX where you are recommended to use **fort77** rather than **f77** since it allows you to specify libraries in a way which is compatible with all the other platforms. For AIX on the RS/6000 the Fortran compiler is called **xlf**, but in more recent versions of AIX the name **f77** can also be used.

The table below shows the minimum command that should be used for compiling and linking in the CERN environment.

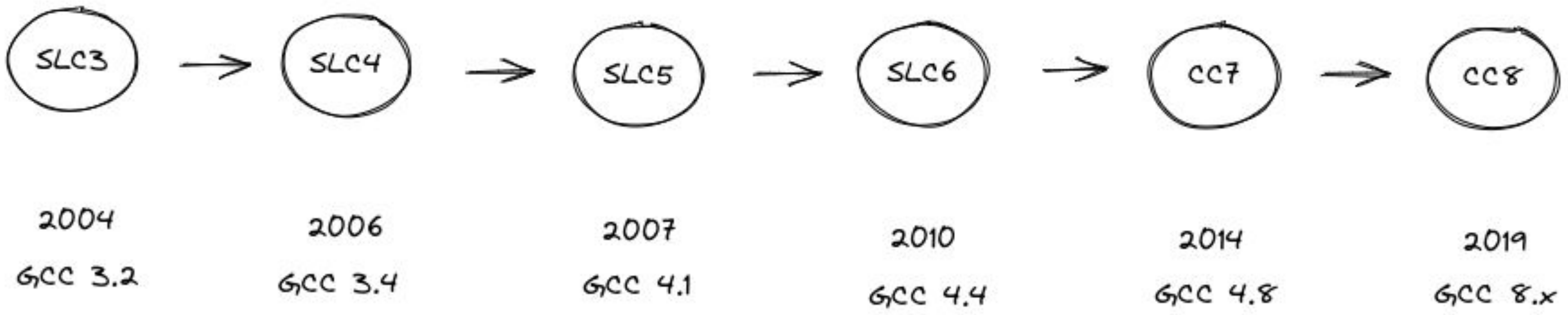
Machine	Compilation only	Compiling and/or Linking
IBM/AIX	xlf -c -qextname	xlf -qextname
HP/UX	fort77 -c +ppu	fort77 +ppu
Others	f77 -c	f77

As we saw in the section above, by default Unix compilers generate an executable module. The option "**-c**" (compile only) generates an object file but causes the linking phase to be surpressed.

The options -qextname on AIX and +ppu on HP/UX are explained in "Compiling and Linking Options" on page 50 and are ESSENTIAL for compatibility with the CERN Program Library.

CERN Unix User Guide, September 2002 https://cds.cern.ch/record/291184/files/cer-000212302.pdf

Linux Era


SLC3
SLC4
SLC5
SLC6
CC7
CC8
2004
GCC 3.2
2006
GCC 3.4
2007
GCC 4.1
2010
GCC 4.4
2014
GCC 4.8
2019
GCC 8.x

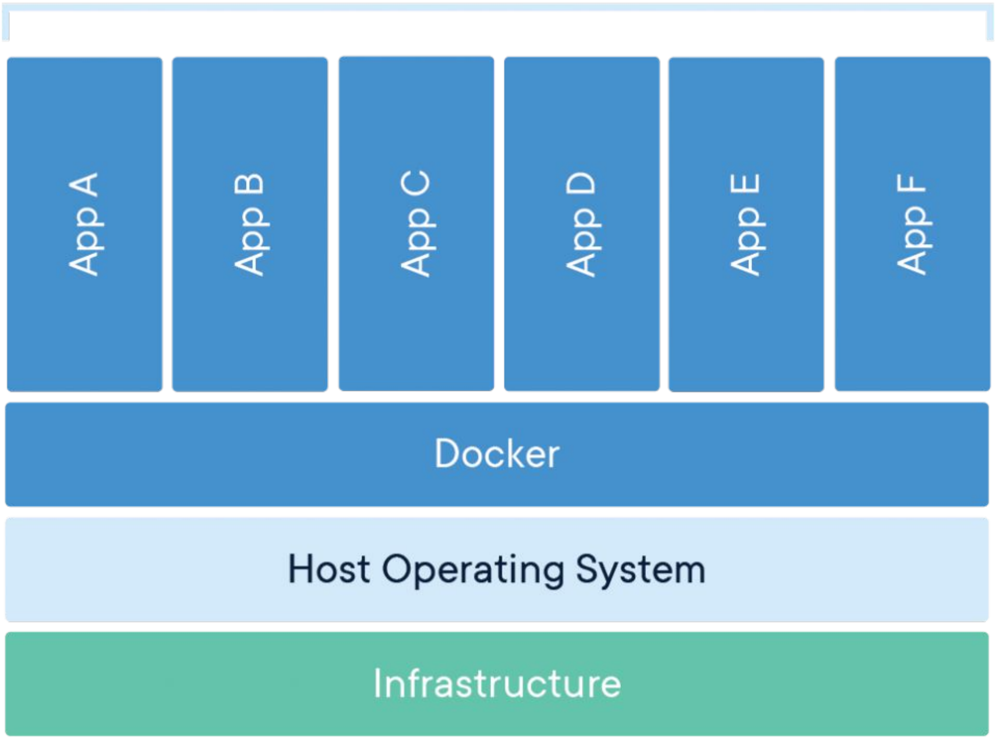
And we could go on...

Will i still be able to access my data?

Will my data format still be readable?

...

Containerized Applications



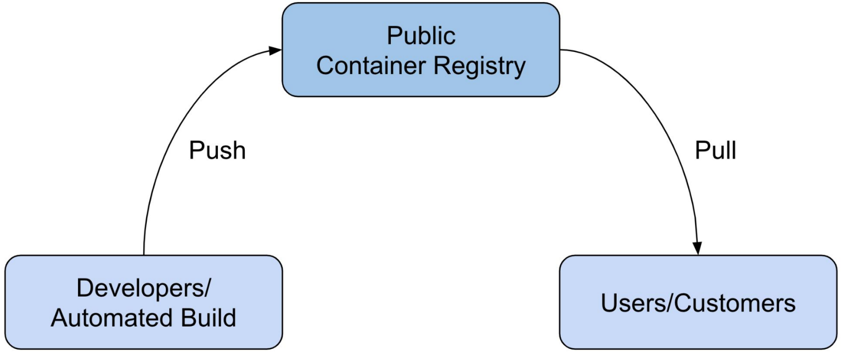
Evolve infrastructure and applications separately

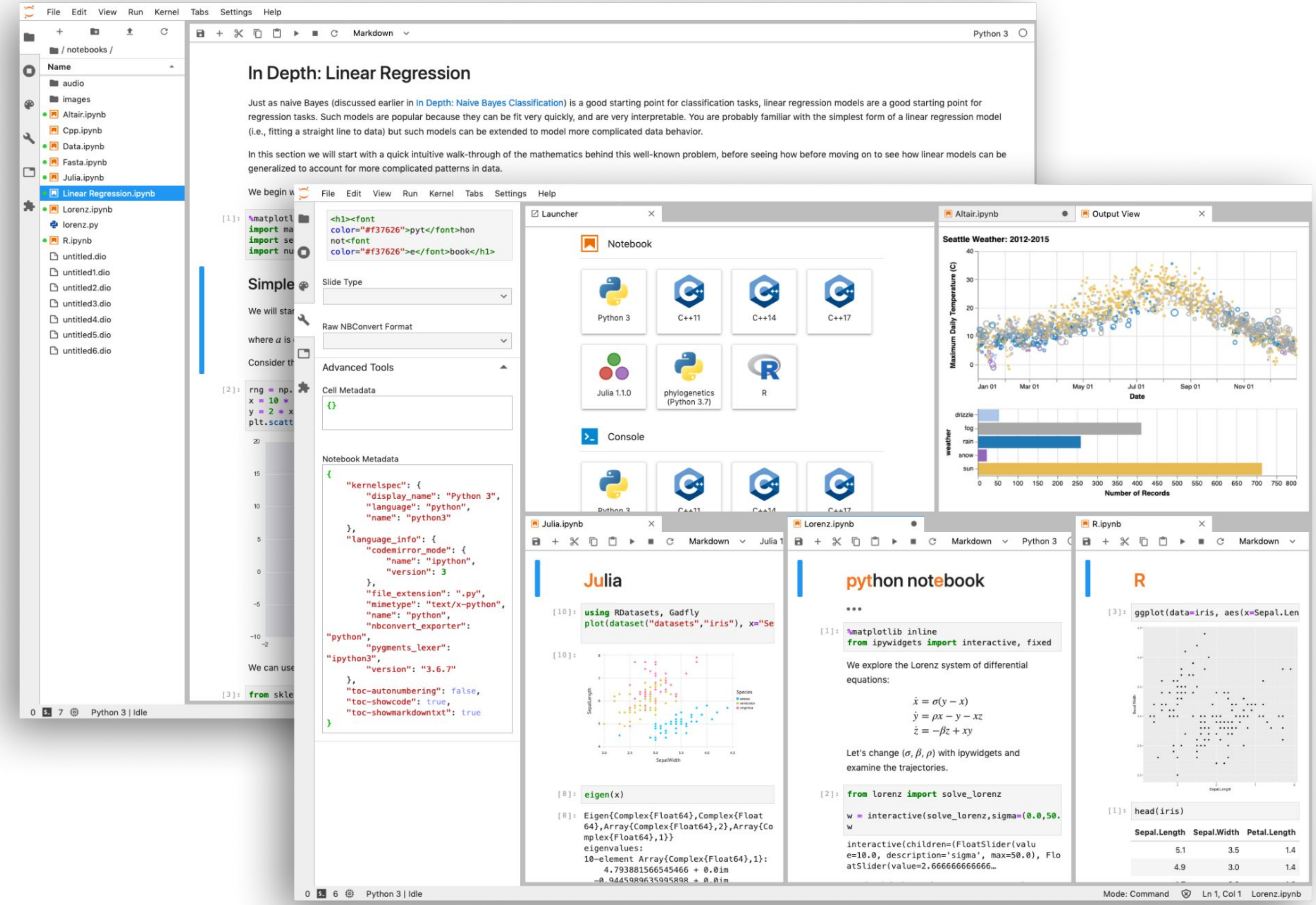
Shared kernel, but distinct *operating systems*

Immutable, tagged images

Sharing using private and public registries

```
# Base image with explicit versioning
#
# Can be different from the host OS running the container,
# which could be running CentOS or other distribution
FROM ubuntu:16.04

# Trackback to image maintainer
MAINTAINER ricardo.rocha@cern.ch

# Dependencies with explicit versioning
RUN apt-get update
RUN apt-get install -y gcc:7.5.0 wget:1.12
RUN pip install scipy:0.18.1

# Any custom files, binaries, even data can be added
COPY ./specialfile /

# Anything else can be run as part of the image build
RUN wget http://domain/customscript -O /run.sh

# Default command being run on start
CMD ["/run.sh"]
```

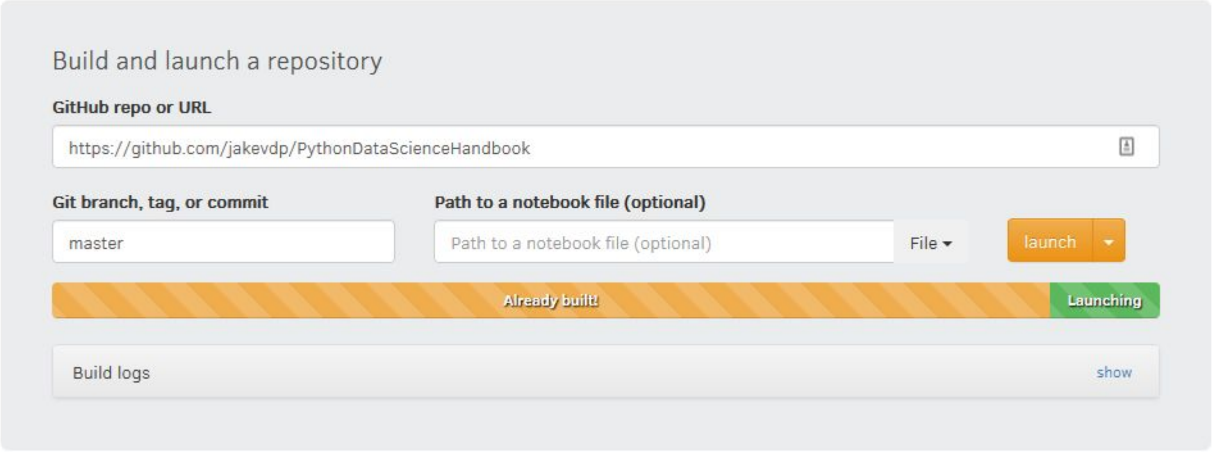


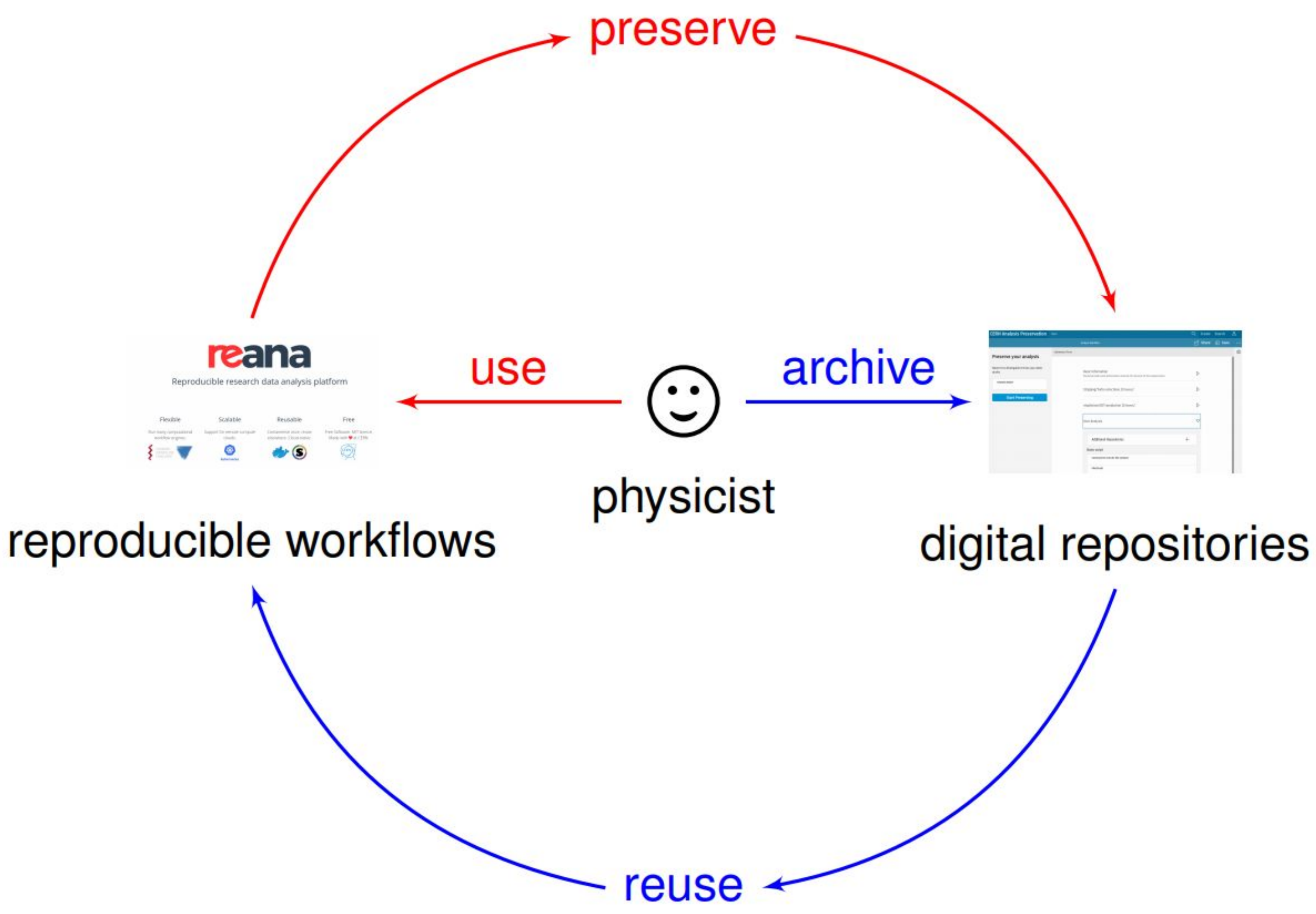

File Edit View Run Kernel Tabs Settings Help
/ notebooks /
Name
audio
images
Altair.ipynb
Cpp.ipynb
Data.ipynb
Fasta.ipynb
Julia.ipynb
Linear Regression.ipynb
Lorenz.ipynb
lorenz.py
R.ipynb
untitled.dio
untitled1.dio
untitled2.dio
untitled3.dio
untitled4.dio
untitled5.dio
untitled6.dio
Markdown
Python 3
In Depth: Linear Regression
Just as naive Bayes (discussed earlier in In Depth: Naive Bayes Classification) is a good starting point for classification tasks, linear regression models are a good starting point for regression tasks. Such models are popular because they can be fit very quickly, and are very interpretable. You are probably familiar with the simplest form of a linear regression model (i.e., fitting a straight line to data) but such models can be extended to model more complicated data behavior.
In this section we will start with a quick intuitive walk-through of the mathematics behind this well-known problem, before seeing how before moving on to see how linear models can be generalized to account for more complicated patterns in data.
Python 3 | Idle
Slide Type
Raw NBConvert Format
Advanced Tools
Cell Metadata
Notebook Metadata
Launcher
Notebook
Python 3
C++11
C++14
C++17
Julia 1.1.0
phylogenetics (Python 3.7)
R
Console
Altair.ipynb
Output View
Seattle Weather: 2012-2015
Maximum Daily Temperature (C)
Date
weather
Number of Records
Julia.ipynb
Julia
using RDatasets, Gadfly
eigen(x)
Lorenz.ipynb
python notebook
%matplotlib inline
from ipywidgets import interactive, fixed
We explore the Lorenz system of differential equations:
Let's change (σ, β, ρ) with ipywidgets and examine the trajectories.
from lorenz import solve_lorenz
R.ipynb
R
head(iris)
Sepal.Length
Sepal.Width
Petal.Length
Mode: Command
Ln 1, Col 1
Lorenz.ipynb

binder
(beta)

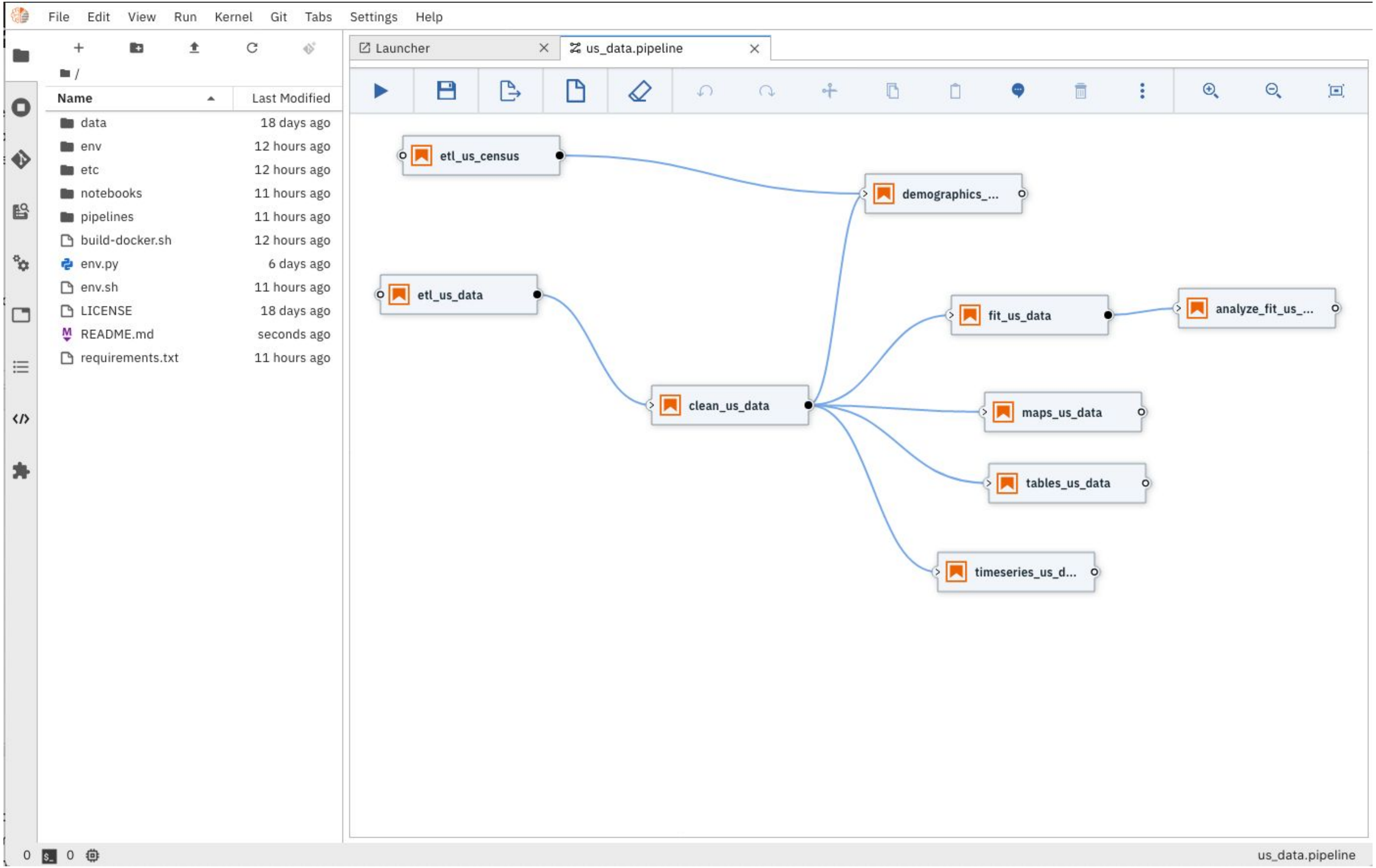
Turn a GitHub repo into a collection of interactive notebooks

Have a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.





preserve
reana
Reproducible research data analysis platform
Flexible
Scalable
Reusable
Free
use
archive
physicist
reproducible workflows
digital repositories
reuse



File
Edit
View
Run
Kernel
Git
Tabs
Settings
Help
Launcher
us_data.pipeline
Name
Last Modified
data
18 days ago
env
12 hours ago
etc
12 hours ago
notebooks
11 hours ago
pipelines
11 hours ago
build-docker.sh
12 hours ago
env.py
6 days ago
env.sh
11 hours ago
LICENSE
18 days ago
README.md
seconds ago
requirements.txt
11 hours ago
etl_us_census
demographics_...
etl_us_data
fit_us_data
analyze_fit_us_...
clean_us_data
maps_us_data
tables_us_data
timeseries_us_d...
us_data.pipeline

Containers & Open Data in Science

CONTAINERS IN THE CLOUD

Standardized platforms allow researchers to run each other's software – no installation required. **By Jeffrey M. Perkel**

Murphy's law for the digital age: anything that can go wrong, will go wrong during a live demonstration. For Ben Marwick, that happened in front of a roomful of landscape-archaeology students in Berlin. The topic: computational reproducibility using Docker.

Docker is a software tool that generates 'containers' – standardized computational environments that can be shared and reused. Containers ensure that computational analyses always run on the same underlying infrastructure, fostering reproducibility. Docker thereby insulates researchers from the challenges of installing and updating research software. However, it can be difficult to use.

Marwick, an archaeologist at the University of Washington in Seattle, had become proficient in migrating Docker configuration files ('Dockerfiles') from one project to the next, making minor tweaks and getting them to work. Colleagues in Germany invited him to teach their students how to follow suit. But because every student had a slightly different set of hardware and software installed, each one required a customized configuration. The demo "was a complete disaster", Marwick says.

Today, a growing collection of services allows researchers to sidestep such confusion. Using these services – which include Binder, Code Ocean, Colaboratory, Gigantum and Nextjournal – researchers can run code in the cloud without needing to install more software. They can lock down their software configurations, migrate those environments from laptops to high-performance computing clusters and share them with colleagues. Educators can create and share course materials with students, and journals can improve the reproducibility of results in published articles. It's never been easier to understand, evaluate, adopt and adapt the computational methods on which modern science depends.

William Coon, a sleep researcher at Harvard Medical School in Boston, Massachusetts, spent weeks writing and debugging an algorithm, only to discover that a colleague's containerized code could have saved a lot of time. "I could have just gotten up and running, using all of the debugging work that he had already done, at the click of a button," he says.

Scientific software often requires installing, navigating and troubleshooting a byzantine network of computational 'dependencies'

– the code libraries and tools on which each software module relies. Some have to be compiled from source code or configured just so, and an installation that should take a few minutes can degenerate into a frustrating online odyssey through websites such as Stack Overflow and GitHub. "One of the hardest parts of reproducibility is getting your computer set up in exactly the same way as somebody else's computer is set up. That is just ridiculously difficult," says Kirstie Whitaker, a neuroscientist at the Alan Turing Institute in London.

Easier evaluation

Docker reduces that to a single command. "Docker really provides reduced friction for that stage of the cycle of reproducing somebody else's work, in which you have to build the software from source and combine it with other external libraries," says Lorena Barba, a mechanical and aerospace engineer at George Washington University in Washington DC. "It facilitates that part, making it less error-prone, making it less onerous in researcher time."

Barba's team does most of its work in Docker containers. But that is a computationally savvy research group; others might find the process daunting. A text-based 'command-line' application, Docker has dozens of options, and building a working Dockerfile can be an exercise in frustration.

That's where the cloud-based services come in. Binder is an open-source project that allows users to test-drive computational notebooks – documents such as Jupyter or R Markdown notebooks, which blend code, figures and text. Colaboratory (free), Code Ocean, Gigantum and Nextjournal (the latter three have free and paid tiers) let users write code in the cloud as well and, in some cases, bundle it with the data to be processed. These platforms also allow users to modify the code and apply it to other data sets, and provide version-control features for reviewing changes.

Such tools make it easier for researchers to evaluate their colleagues' work. "With Binder, you have taken that barrier [of software installation] away," says Karthik Ram, a computational ecologist at the University of California, Berkeley. "If I can click that button, be dropped into a notebook where everything is installed, the environment is exactly the way you intended it to be, then you've made my life easier to go take a look and give you feedback."

Identifying required dependencies, and where to find them, varies with the platform. On Code Ocean and Gigantum, it's a point-and-click operation, whereas Binder requires a list of dependencies in a Github respository. Whitaker's advice: codify your computing environment as early as possible in a project, and stick with it. "If you try and do it at the end, then you are basically doing archaeology on your code, and it's really, really hard," she says. Ram developed a tool called Holepunch for projects that use the statistical programming language R. Holepunch distils the process of setting up Binder into four simple commands. (See examples of our code running on all five platforms at go.nature.com/2ps9se1.)

The easiest way to try Binder is at mybinder.org, a free, albeit computationally limited, website. Or, for greater power and security, researchers can build private 'BinderHubs' instead. The Alan Turing Institute has two, including one called Hub23 (a reference to Hut 23 at the Second World War code-breaking facility at Bletchley Park, UK), that provides greater computational resources and the ability to work with data sets that cannot be publicly shared, Whitaker says. The Pangeo community, which promotes open, reproducible and scalable geoscience, built a dedicated BinderHub so that researchers can explore climate-modelling and satellite data sets that can amount to tens of terabytes, says Joe Hamman, a computational hydroclimatologist at the National Center for Atmospheric Research in Boulder, Colorado. (Whitaker's team has published a tutorial on building a BinderHub at go.nature.com/349jscv.)

"Researchers can be confident that their code will remain usable, whichever platform they choose."

Languages and clouds

Google's Colaboratory is basically a cross between a Jupyter notebook and Google Docs, meaning users can share, comment on and jointly edit notebooks, which are stored on Google Drive. Users execute their code in the Google cloud – only the Python language is officially supported – on a standard central processing unit (CPU), a graphics processing unit (GPU) or a tensor processing unit (TPU), a specialized chip optimized for Google's TensorFlow deep-learning software. "You can open up your notebook or someone else's notebook from GitHub, start playing around with it and then save your copy on Google Drive and work on it later," says Jake VanderPlas, a member of the Colaboratory team at Google in Seattle.

Nextjournal supports notebooks written in Python, R, Julia, Bash and Clojure, with more languages in development. According to Martin Kavalar, chief executive of Nextjournal, which is based in Berlin, the company has registered nearly 3,000 users since it launched the platform on 8 May.

Gigantum, a beta version of which launched last year, features a browser-based client that users can install on their own system or remotely, for cloud-based coding and execution in the Jupyter and RStudio coding environments. Coon, who uses Gigantum to run machine-learning algorithms in the Amazon cloud, says the service makes it easy for collaborators to hit the ground running. "[They] can read through my Gigantum notebooks and use this cloud-compute infrastructure to do the training and learning," he explains.

Then there's Code Ocean, which supports both notebooks and conventional scripts in Python, R, Julia, Matlab and C, among other languages. Several journals now use Code Ocean for peer review and to promote computational reproducibility, including titles from Taylor & Francis, De Gruyter and SPIE. In 2018, *Nature Biotechnology*, *Nature Machine Intelligence* and *Nature Methods* launched a pilot programme to use Code Ocean for peer review; *Nature*, *Nature Protocols* and *BMC Bioinformatics* subsequently joined the trial. More than 95 papers have now been involved in the trial, according to Erika Pastrana, editorial director of Nature Research's applied-science and chemistry journals, and more than 20 of those have been published.

Felicity Allen, a computer scientist at the Wellcome Sanger Institute in Hinxton, UK, co-authored one study in that trial, which analysed the types of mutation that can arise from CRISPR-based gene editing (F. Allen *et al. Nature Biotechnol.* **37**, 64–72; 2019). She estimates that it took a week to get the Code Ocean environment working. "The reviewers seemed to really like it," Allen says. "And I think it was really nice that it made an example that someone could just press 'go' on and it would run."

Although some worry about the long-term viability of commercial container-computing services, researchers do have options. Simon Adar, chief executive of Code Ocean, notes that Code Ocean 'compute capsules' are archived by the CLOCKSS project, which preserves digital copies of online scientific literature. And Code Ocean, Gigantum and Nextjournal allow Dockerfiles to be exported for use on other platforms. All of which means that researchers can be confident that their code will remain usable, whichever platform they choose.

Benjamin Haibe-Kains, a computational pharmacogenomics researcher at the Princess Margaret Cancer Centre in Toronto, Canada, adopted Code Ocean to respond quickly to critiques of an analysis he published in *Nature* (B. Haibe-Kains *et al. Nature* **504**, 389–393; 2013). For him, Code Ocean provides a way to ensure his code can be used and evaluated by his team, peer reviewers and the broader scientific community. "It's not so much that an analysis must be correct or wrong," he says. "Nothing is really fully correct in this world. However, if you're very transparent about it, you can always communicate efficiently in the face of criticism. You have nothing to hide: everything is there."

Jeffrey M. Perkel is technology [illegible] *Nature*.

nature

nature physics

PERSPECTIVE

https://doi.org/10.1038/s41567-018-0342-2

Corrected: Publisher Correction

OPEN

Open is not enough

Xiaoli Chen[1,2], Sünje Dallmeier-Tiessen[1*], Robin Dasler[1,11], Sebastian Feger[1,3], Pamfilos Fokianos[1], Jose Benito Gonzalez[1], Harri Hirvonsalo[1,4,12], Dinos Kousidis[1], Artemis Lavasa[1], Salvatore Mele[1], Diego Rodriguez Rodriguez[1], Tibor Šimko[1*], Tim Smith[1], Ana Trisovic[1,5*], Anna Trzcinska[1], Ioannis Tsanaktsidis[1], Markus Zimmermann[1], Kyle Cranmer[6], Lukas Heinrich[6], Gordon Watts[7], Michael Hildreth[8], Lara Lloret Iglesias[9], Kati Lassila-Perini[4] and Sebastian Neubert[10]

The solutions adopted by the high-energy physics community to foster reproducible research are examples of best practices that could be embraced more widely. This first experience suggests that reproducibility requires going beyond openness.

Open science and reproducible research have become pervasive goals across research communities, political circles and funding bodies[1–3]. The understanding is that open and reproducible research practices enable scientific reuse, accelerating future projects and discoveries in any discipline. In the struggle to take concrete steps in pursuit of these aims there has been much discussion and awareness-raising, often accompanied by a push to make research products and scientific results open quickly.

Although these are laudable and necessary first steps, they are not sufficient to bring about the transformation that would allow us to reap the benefits of open and reproducible research. It is time to move beyond the rhetoric and the trust in quick fixes and start designing and implementing tools to power a more profound change.

Our own experience from opening up vast volumes of data is that openness cannot simply be tacked on as an afterthought at the end of the scientific endeavour. In addition, openness alone does not guarantee reproducibility or reusability, so it should not be pursued as a goal in itself. Focusing on data is also not enough: it needs to be accompanied by software, workflows and explanations, all of which need to be captured throughout the usual iterative and closed research lifecycle, ready for a timely open release with the results.

Thus, we argue that having the reuse of research results as a goal requires the adoption of new research practices during the data analysis process. Such practices need to be tailored to the needs of each given discipline with its particular research environment, culture and idiosyncrasies. Services and tools should be developed with the idea of meshing seamlessly with existing research procedures, encouraging the pursuit of reusability as a natural part of researchers' daily work (Fig. 1). In this way, the generated research products are more likely to be useful when shared openly.

In tackling the challenge of enabling reusable research, we keep these ideas as our guiding light when putting changes into practice in our community—high-energy physics (HEP). Here, we illustrate our approach, particularly through our work at CERN, and present our community's requirements and rationale. We hope that the explanation of our challenges and solutions will stimulate discussions around the practical implementation of workflows for reproducible and reusable research more widely in other scientific disciplines.

Approaching reproducibility and reuse in HEP

To set the stage for the rest of this piece, we first construct a more nuanced spectrum in which to place the various challenges facing HEP, allowing us to better frame our ambitions and solutions. We choose to build on the descriptions introduced by Carole Goble[4] and Lorena A. Barba[5] shown in Table 1.

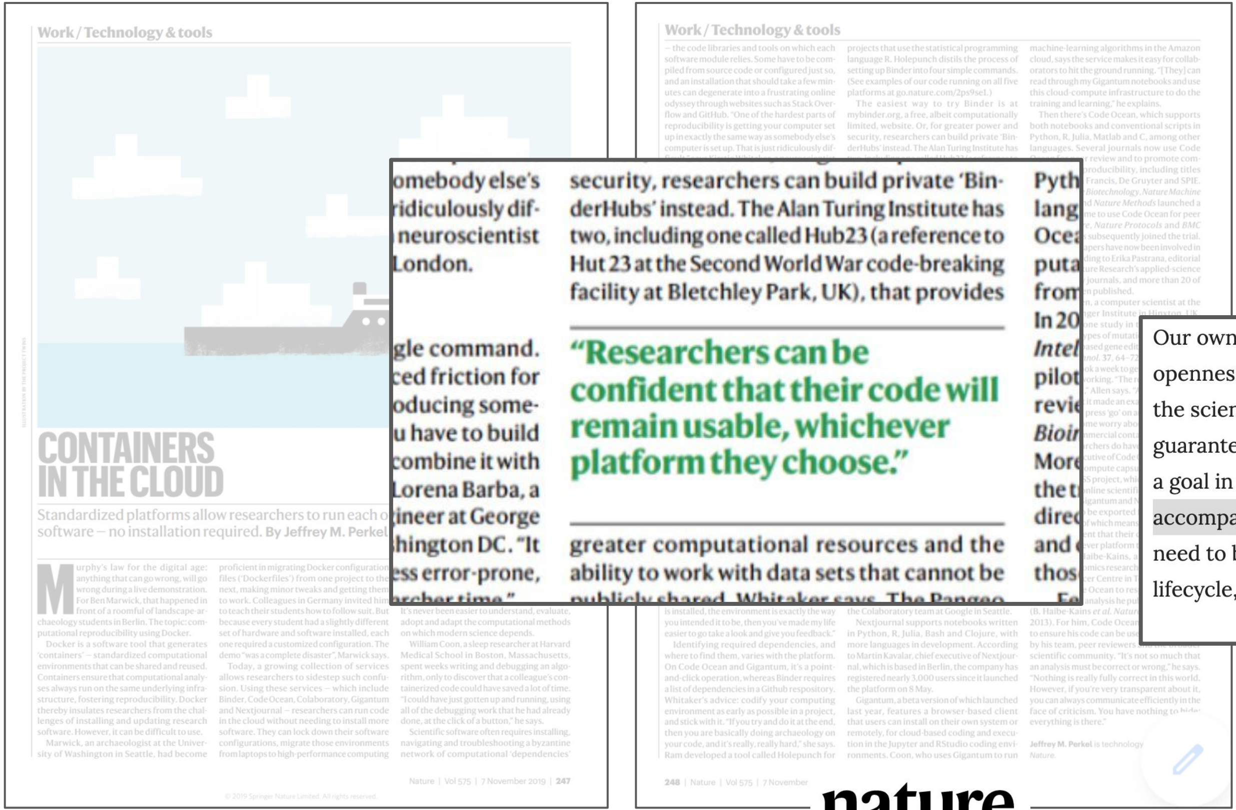
These concepts assume a research environment in which multiple labs have the equipment necessary to duplicate an experiment, which essentially makes the experiments portable. In the particle physics context, however, the immense cost and complexity of the experimental set-up essentially make the independent and complete replication of HEP experiments unfeasible and unhelpful. HEP experiments are set up with unique capabilities, often being the only facility or instrument of their kind in the world; they are also constantly being upgraded to satisfy requirements for higher energy, precision and level of accuracy. The experiments at the Large Hadron Collider (LHC) are prominent examples. It is this uniqueness that makes the experimental data valuable for preservation so that it can be later reused with other measurements for comparison, confirmation or inspiration.

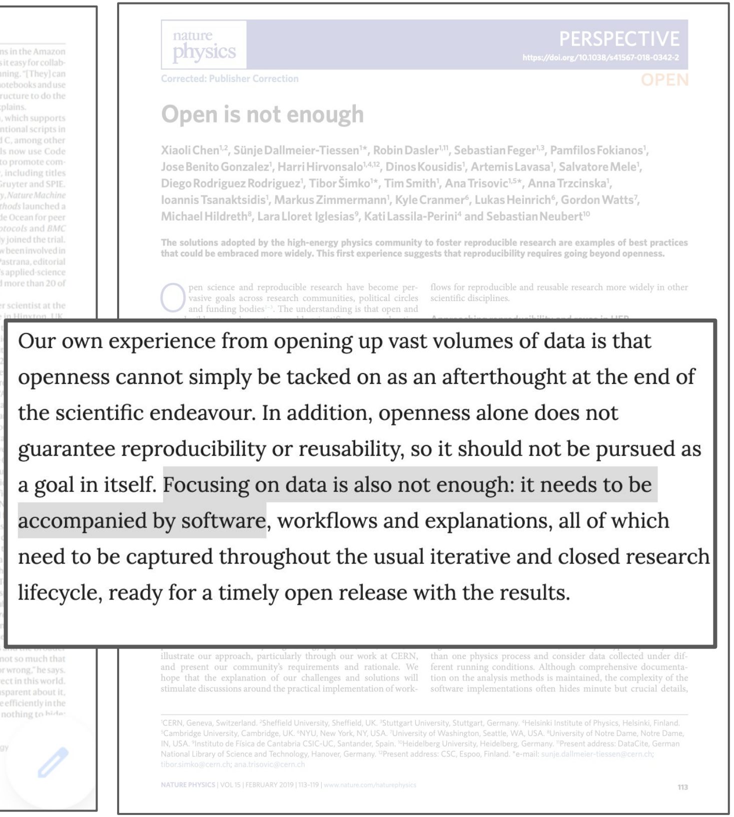
Our considerations here really begin after gathering the data. This means that we are more concerned with repeating or verifying the computational analysis performed over a given dataset rather than with data collection. Therefore, in Table 2 we present a variation of these definitions that takes into account a research environment in which 'experimental set-up' refers to the implementation of a computational analysis of a defined dataset, and a 'lab' can be thought of as an experimental collaboration or an analysis group.

In the case of computational processes, physics analyses themselves are intrinsically complex due to the large data volume and algorithms involved[6]. In addition, the analysts typically study more than one physics process and consider data collected under different running conditions. Although comprehensive documentation on the analysis methods is maintained, the complexity of the software implementations often hides minute but crucial details,

[1]CERN, Geneva, Switzerland. [2]Sheffield University, Sheffield, UK. [3]Stuttgart University, Stuttgart, Germany. [4]Helsinki Institute of Physics, Helsinki, Finland. [5]Cambridge University, Cambridge, UK. [6]NYU, New York, NY, USA. [7]University of Washington, Seattle, WA, USA. [8]University of Notre Dame, Notre Dame, IN, USA. [9]Instituto de Física de Cantabria CSIC-UC, Santander, Spain. [10]Heidelberg University, Heidelberg, Germany. [11]Present address: DataCite, German National Library of Science and Technology, Hanover, Germany. [12]Present address: CSC, Espoo, Finland. *e-mail: sunje.dallmeier-tiessen@cern.ch; tibor.simko@cern.ch; ana.trisovic@cern.ch

Containers & (Open) Science





nature

opendata CERN

Help About

Explore more than **two petabytes** of open data from particle physics!

Start typing... Search

search examples: collision datasets, keywords:education, energy:7TeV

Explore

datasets

software

environments

documentation

Focus on

ATLAS

ALICE

CMS

LHCb

OPERA

PHENIX

Data Science

Get started

Can we also build on this to scale out our analysis?

Kubernetes

Spun out of Google as an open source container orchestration project

Built on lessons from Borg and Omega

Borg, Omega, and Kubernetes

LESSONS LEARNED FROM THREE CONTAINER-MANAGEMENT SYSTEMS OVER A DECADE

BRENDAN BURNS, BRIAN GRANT, DAVID OPPENHEIMER, ERIC BREWER, AND JOHN WILKES, GOOGLE INC.

Though widespread interest in software containers is a relatively recent phenomenon, at Google we have been managing Linux containers at scale for more than ten years and built three different container-management systems in that time. Each system was heavily

Loosely coupled collection of components to deploy, maintain and scale workloads

Declarative, Load Balancing, Self Healing, Auto Scaling

Service and Batch Workloads

Kubernetes

Largest open source project after kernel

- **45.000** contributors, **148.000** code commits
- **83.000** pull requests, **1.1M** contributions
- **2000+** contributing companies

Google, RedHat, VMware, Huawei, Microsoft, IBM, Fujitsu, …

Open community welcome to contributions

Special Interest Groups (SIGs) : Auto-Scaling, Multi-Cluster, Scheduling, ...

Largely used both in Research and Industry

Borg, Omega, and Kubernetes

LESSONS LEARNED FROM THREE CONTAINER-MANAGEMENT SYSTEMS OVER A DECADE

BRENDAN BURNS, BRIAN GRANT, DAVID OPPENHEIMER, ERIC BREWER, AND JOHN WILKES, GOOGLE INC.

Though widespread interest in software containers is a relatively recent phenomenon, at Google we have been managing Linux containers at scale for more than ten years and built three different container-management systems in that time. Each system was heavily

Kubernetes

Lingua franca of the cloud

Managed services offered by all major public clouds

Multiple options for on-premise or self-managed deployments

Common declarative API for basic infrastructure : compute, storage, networking

Healthy ecosystem of tools offering extended functionality

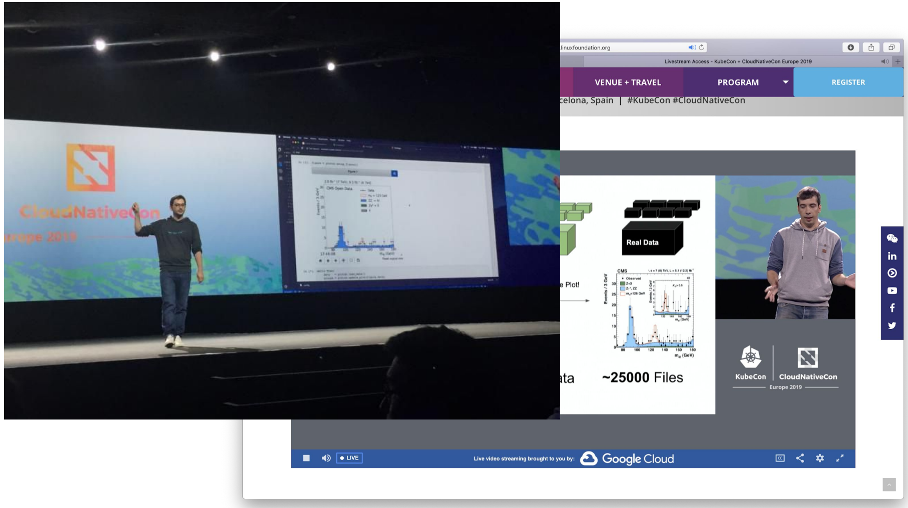
Rediscovering the Higgs

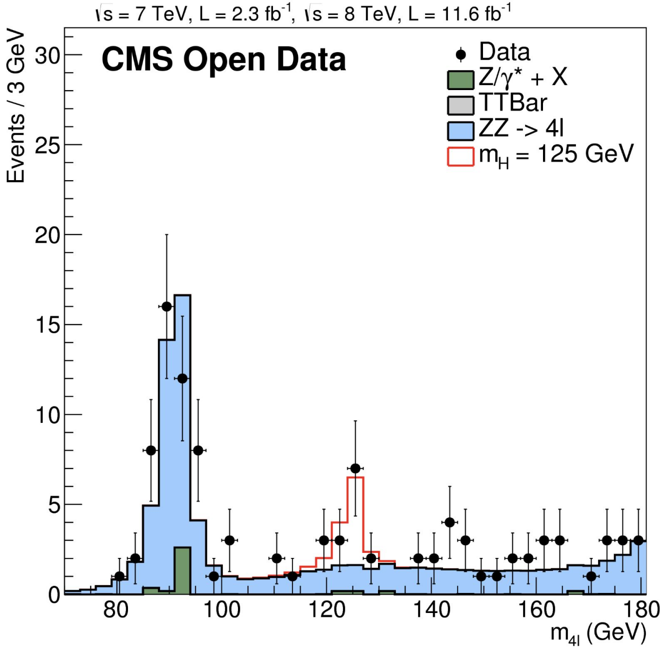
Like it's 2019...

Challenge: H→4l re-discovery on CMS Open Data

Benchmark analysis based on Open LHC Data.

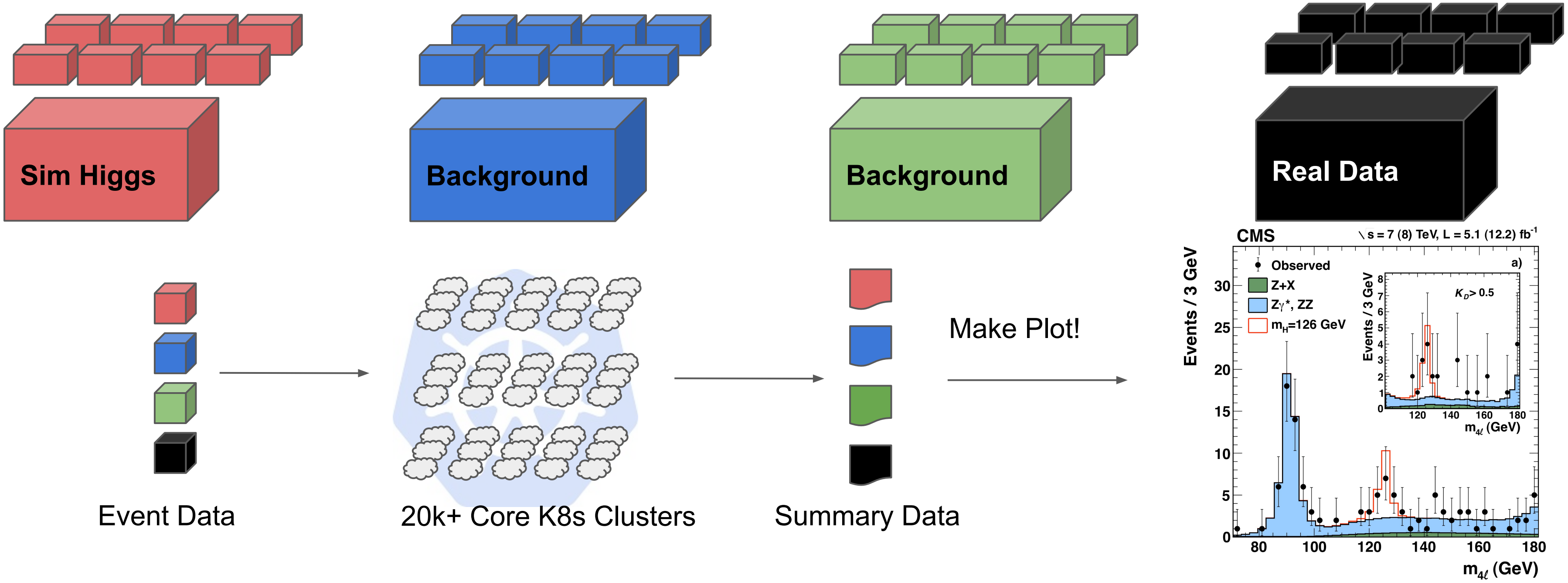
Goal: Fit it within a live demo for 20-minute [Keynote at KubeCon EU 2019](#)
Learn something about cloud-native analysis, reproducibility, Open Data.
Have some Fun.





Challenge: H→4l re-discovery on CMS Open Data

what would this look like in a cloud-native approach?



70 TB of Physics Data **~25000** Files

```
[16:01:21] cmsusr@e6f7bea2253e /Users/lukasheinrich/Code/awesomedemo/higgs-demo/CMSSW_5_3_32/src $ \root -b
  *******************************************
  *                                         *
  *        W E L C O M E  to  R O O T       *
  *                                         *
  *   Version   5.32/00   2 December 2011   *
  *                                         *
  *  You are welcome to visit our Web site  *
  *          http://root.cern.ch            *
  *                                         *
  *******************************************

ROOT 5.32/00 (branches/v5-32-00-patches@42372, Jun 10 2014, 18:26:00 on linuxx8664gcc)

CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
```

cmsopendata/cmssw_5_3_32 ☆

By cmsopendata • Updated 4 months ago

Container

70 TB Dataset

OpenStack Magnum

25000 **Kubernetes Jobs**

Job Results

Interactive Visualization

Aggregation

CERN openlab

Google Cloud

Google Cloud Storage

70 TB Dataset

Cluster on GKE

Max **25000 Cores**

Single Region, 3 Zones

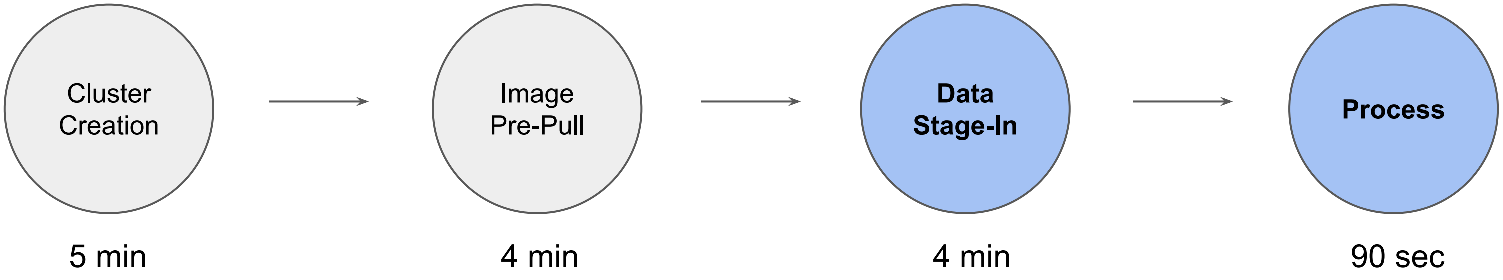
25000 **Kubernetes Jobs**

Job Results

jupyter

Interactive Visualization

Aggregation



vs ~24h for the original analysis

Machine Learning / Kubeflow

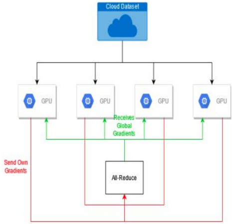
Scale out / distributed training, with CERN OpenLab

Example: Fast Simulation with 3D GANs

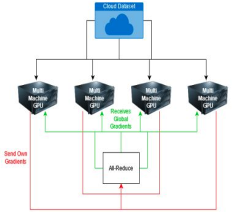
TensorFlow Based

Can benefit from (very) large numbers of GPUs, TPUs

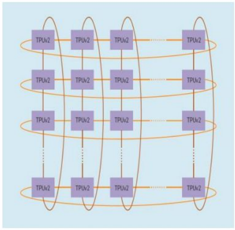
Mirrored Strategy



Multi Worker Mirrored Strategy



TPU Strategy

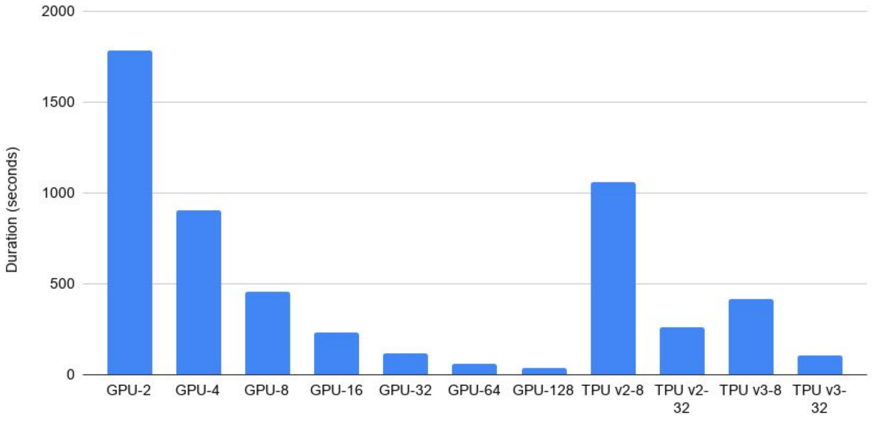


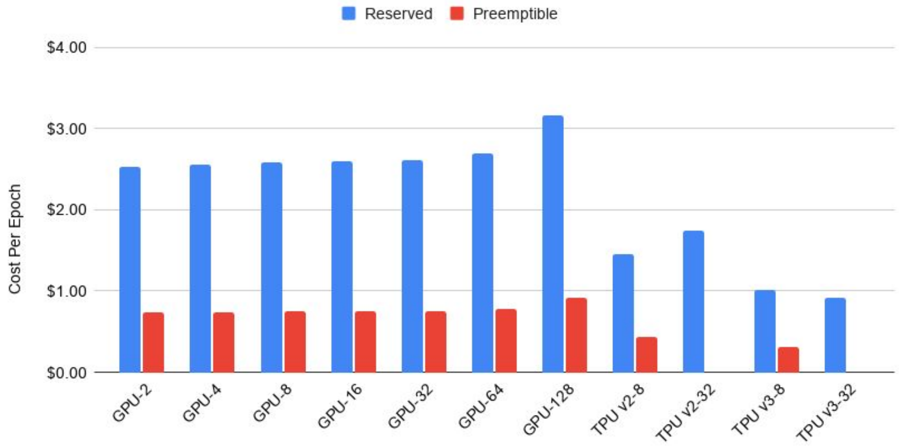
Results using the Google Cloud

From 1 to 128 GPUs: 3550 to 35 seconds per epoch

x100 speedup at the same total cost

TPUs seem to be particularly cost effective





Challenges Remaining

Data Movement

Data Gravity and Egress Costs

Avoiding lock-in to public cloud providers

Bridging with the HPC world

Questions?