Interactive applications on HPC systems

Erich Birngruber
(erich.birngruber@gmi.oeaw.ac.at, @ebirn)
Vienna BioCenter

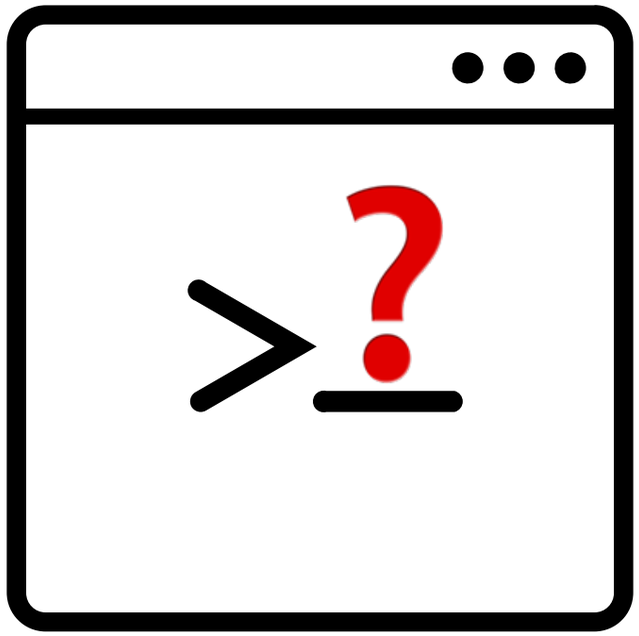
FOSDEM20

Interactive applications on HPC systems

Erich Birngruber
(erich.birngruber@gmi.oeaw.ac.at, @ebirn)
Vienna BioCenter

FOSDEM20

sh$ not good enough?



Usually we submit batch jobs, maybe interactive jobs if-needs-be;
Is the command line good enough? - not always:

* some tools are GUI only, still need major resources
* Interactive data exploration
* Visualizations / plotting
* Collaboration and sharing
* Classroom and training situations
* Analyses triggered by non-HPC users

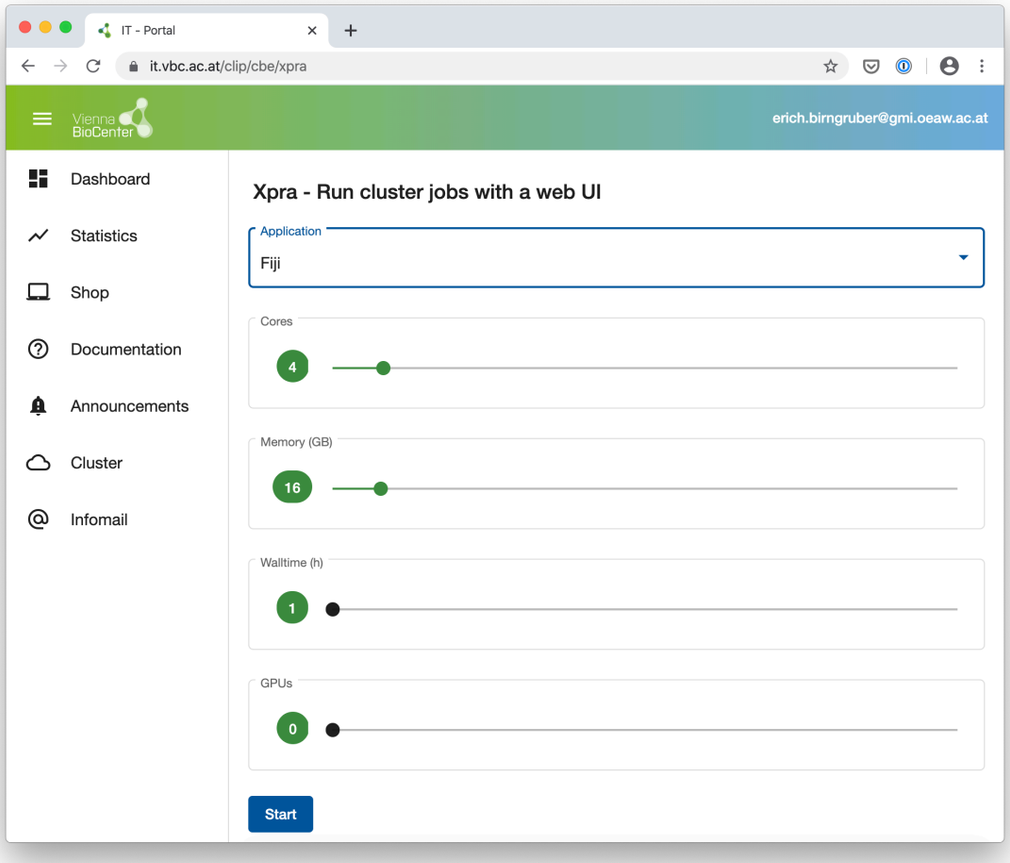
I will bring 4 examples of such applications now.

XPRA

XPRA

- https://xpra.org/
- “screen for X11”
- Allows disconnect / re-connect to existing X sessions
- Web interface for X11 rendering (HTML5 canvas)
- For arbitrary GUI applications
- Containerized in SLURM
- Custom middleware for job management

“Screen”: disconnect from sessions and reconnect later (from somewhere else)
Allows session access via SSH, TCP, and Web!!
Actually: we run X11+canvas client
Drawback: arbitrary GUI apps:
* watch out for keyboard shortcuts (close tab, browser, etc)



Request resources and select application
Applications are launched in Singularity containers (no X11 on compute nodes)

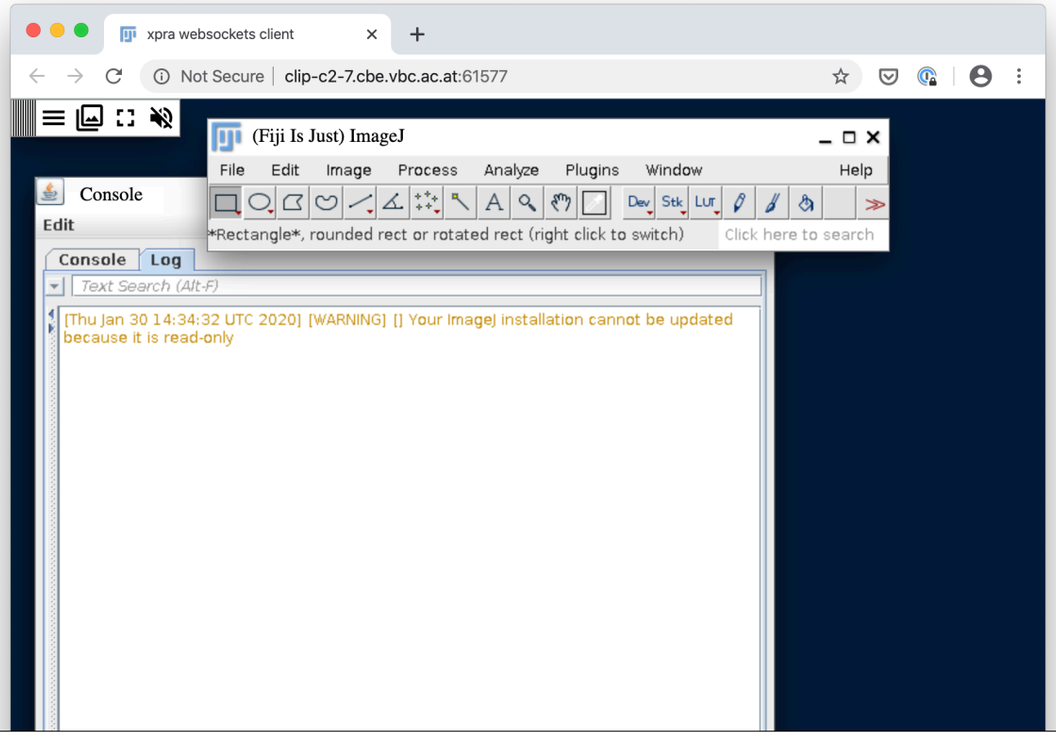
XPRA job submitted

ID	Application	Hours	Cores	Memory (GB)	GPUs	State	Actions
313	Fiji	1	1	4	0	Job running	Join / Settings
56	X-Term	1	12	10	0	Job finished	

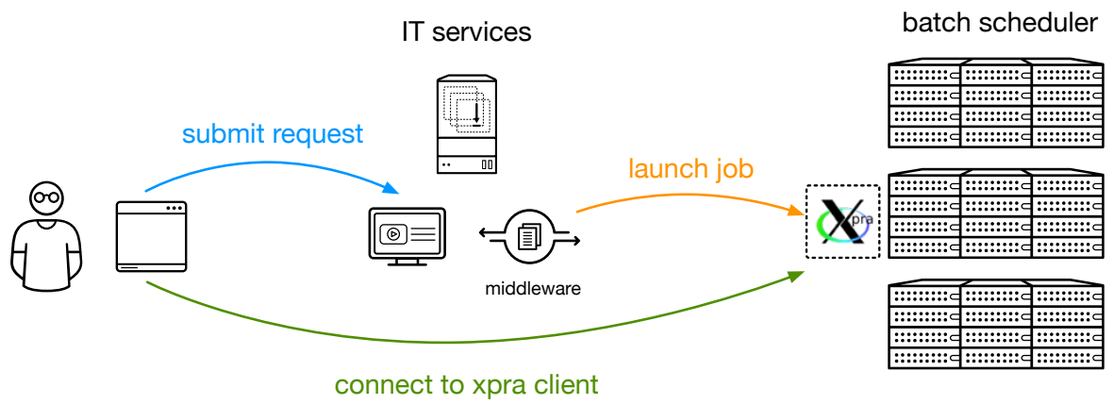
Items per page: 5 1 - 2 of 2

Job will be queued and eventually be ready to connect to

XPRA session



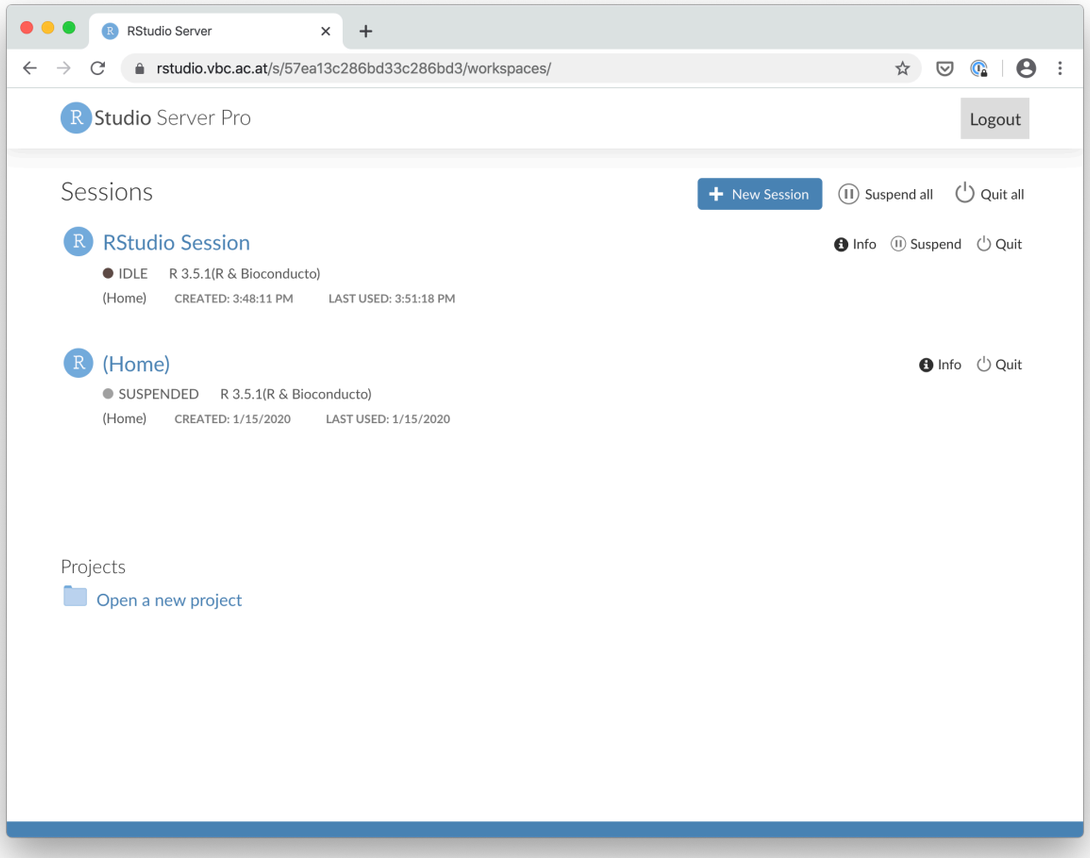
XPRA setup


IT services
batch scheduler
submit request
launch job
middleware
connect to xpra client

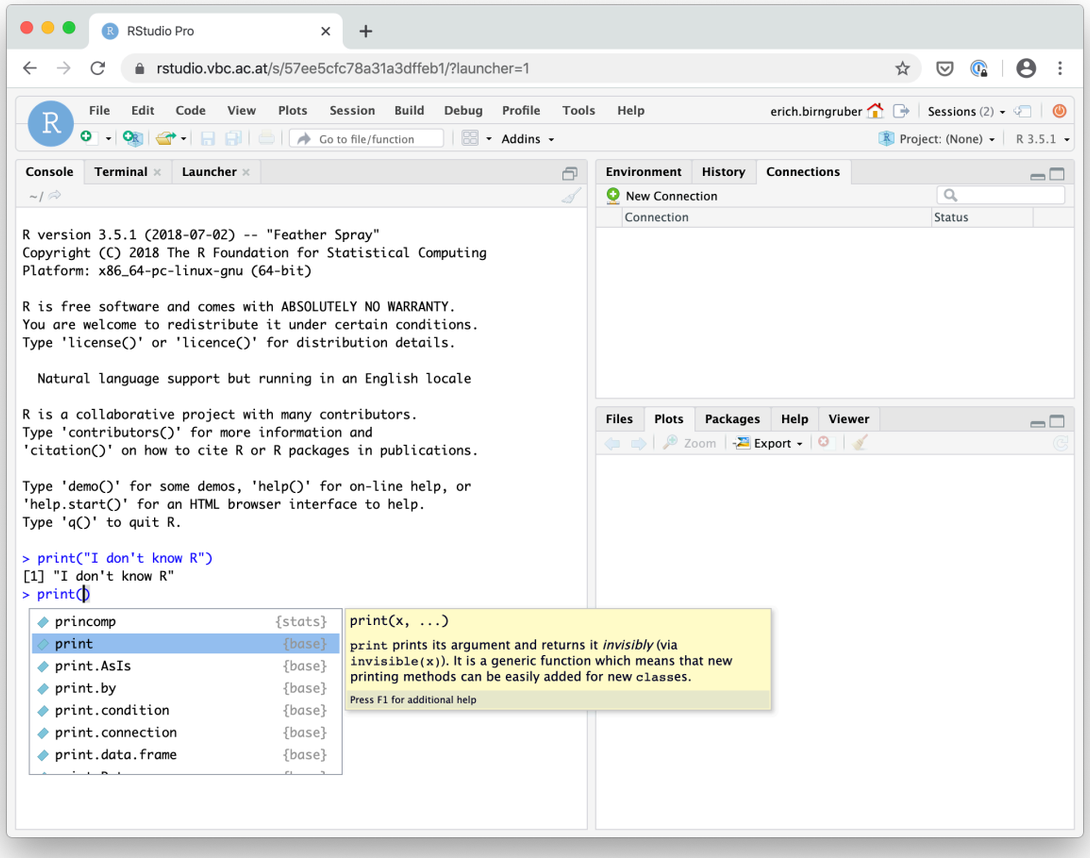
RStudio®

- https://rstudio.com/
- IDE for R language
- Desktop and Web version (RStudio server)
- Commercial version for advanced features
- RStudio company has become a public benefit company
  https://blog.rstudio.com/2020/01/29/rstudio-pbc

Studio Server: launchers = start mechanism for individual session
Commercial features: launchers for various backends (local, Kubernetes, SLURM)



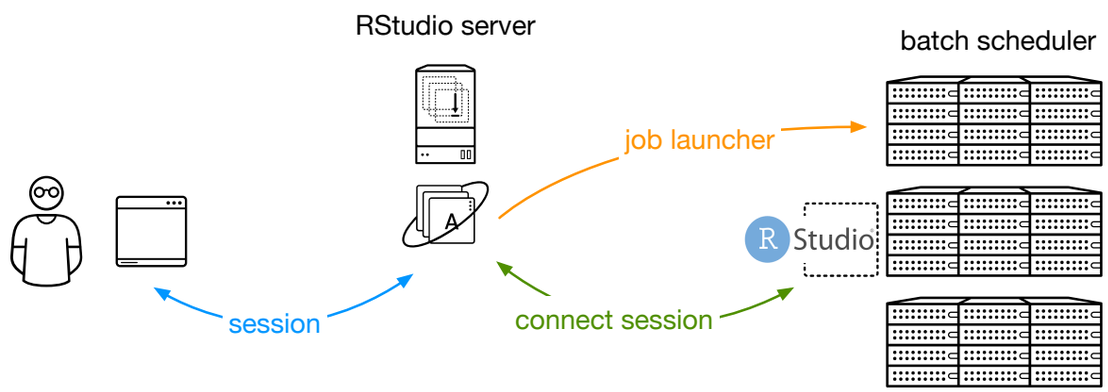
Portal overview
control over multiple sessions



Fully fledged R IDE.

* Interpreter from env modules
* Syntax completion / help
* Launch more jobs from code selection (with different job size than editor session)

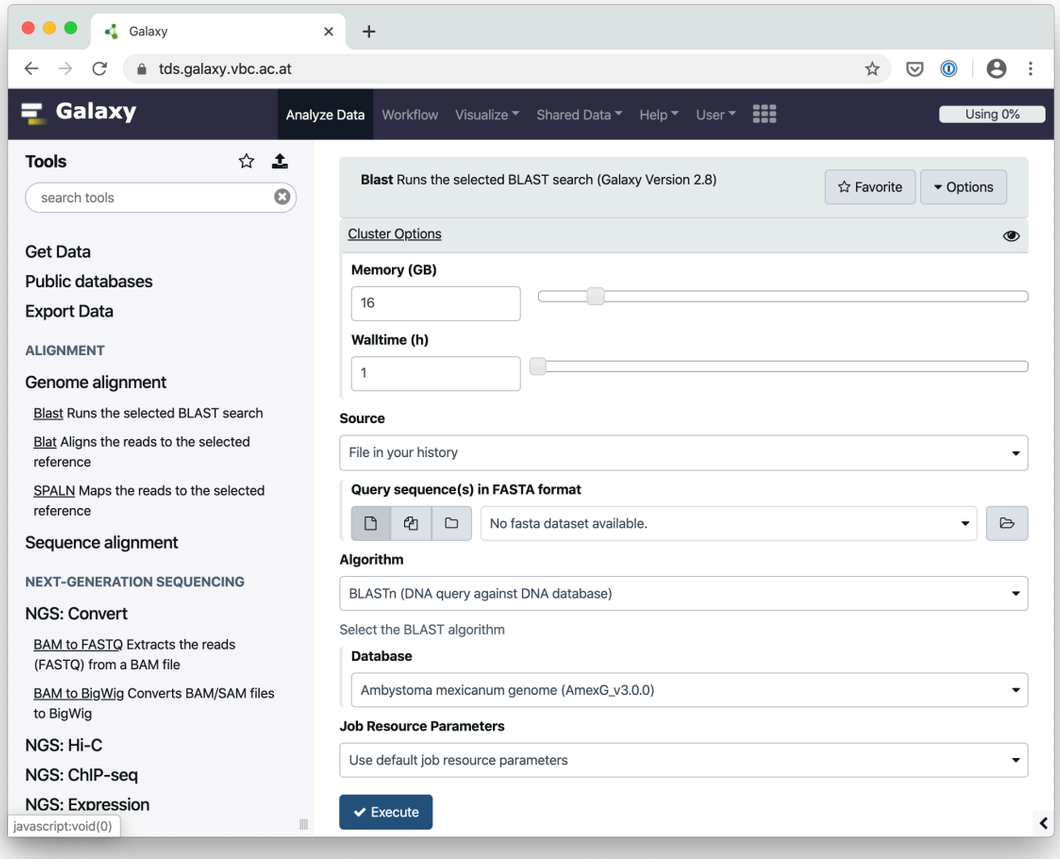
RStudio setup

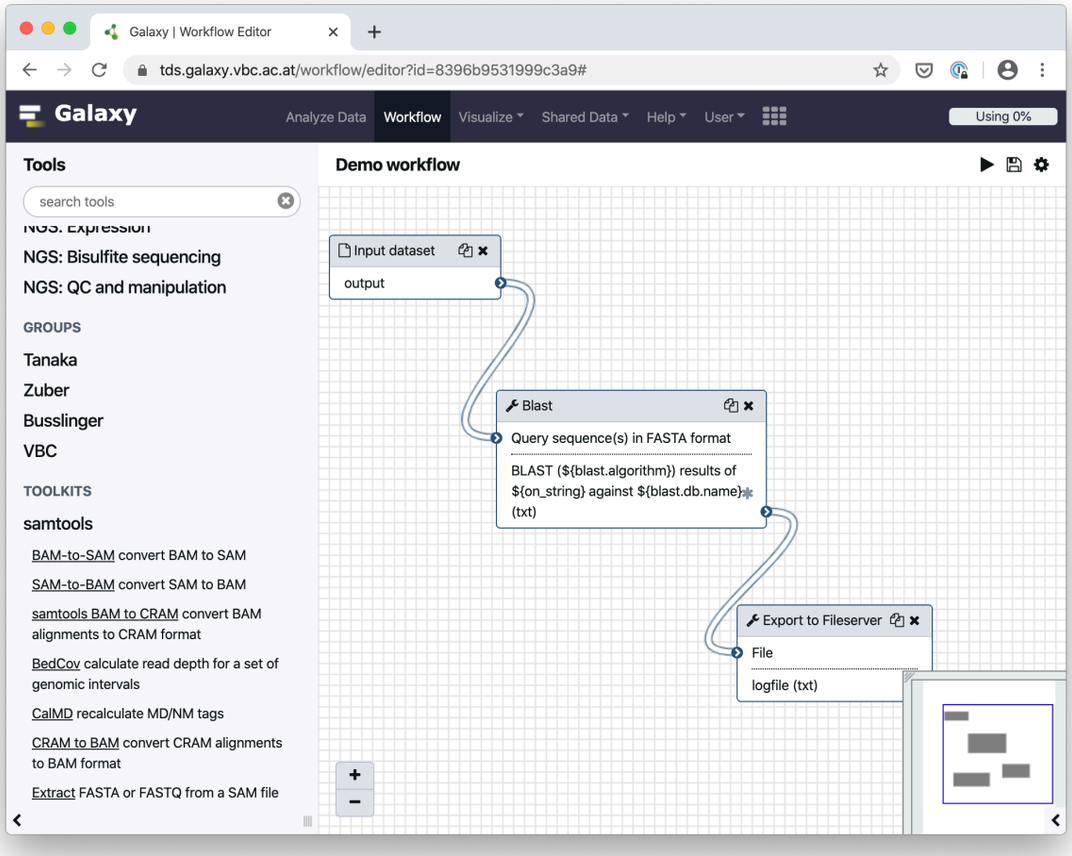


Galaxy
PROJECT

Galaxy
PROJECT

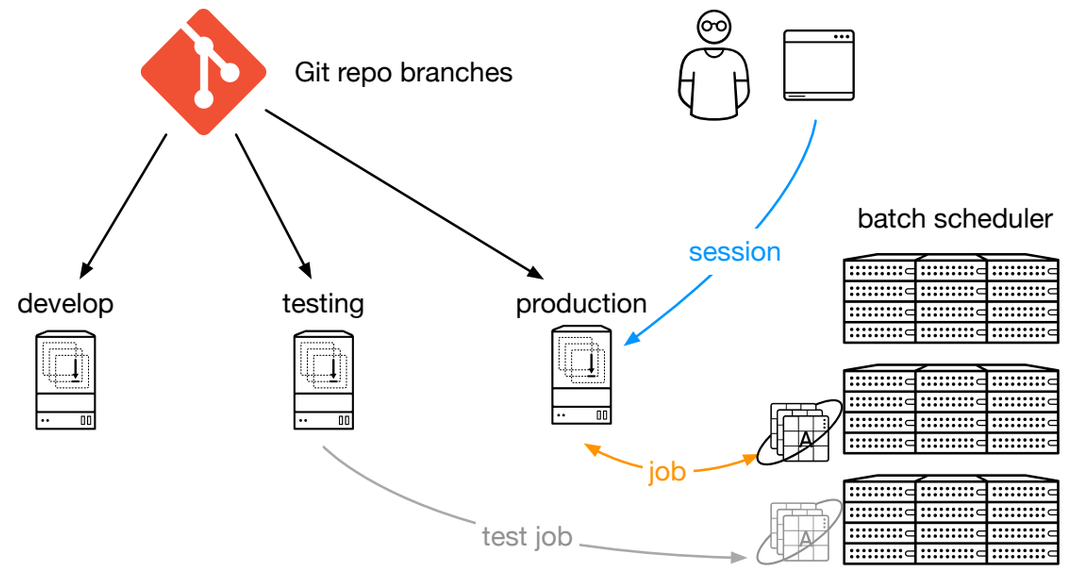
- https://galaxyproject.org/
- Web based workflow tool
- Tools as building blocks (parameters, input, output)
- Tool definitions in XML
- Multiple instances: dev - testing - production


Galaxy
tds.galaxy.vbc.ac.at
Galaxy
Analyze Data
Workflow
Visualize
Shared Data
Help
User
Using 0%
Tools
search tools
Get Data
Public databases
Export Data
ALIGNMENT
Genome alignment
Blast Runs the selected BLAST search
Blat Aligns the reads to the selected reference
SPALN Maps the reads to the selected reference
Sequence alignment
NEXT-GENERATION SEQUENCING
NGS: Convert
BAM to FASTQ Extracts the reads (FASTQ) from a BAM file
BAM to BigWig Converts BAM/SAM files to BigWig
NGS: Hi-C
NGS: ChIP-seq
NGS: Expression
javascript:void(0)
Blast Runs the selected BLAST search (Galaxy Version 2.8)
Favorite
Options
Cluster Options
Memory (GB)
16
Walltime (h)
1
Source
File in your history
Query sequence(s) in FASTA format
No fasta dataset available.
Algorithm
BLASTn (DNA query against DNA database)
Select the BLAST algorithm
Database
Ambystoma mexicanum genome (AmexG_v3.0.0)
Job Resource Parameters
Use default job resource parameters
Execute



Design full workflows via GUI
Requires initial input and starts tools accordingly to do the full pipeline of processing
Bioinformatitions create workflows, can be used for analyses by other users

Galaxy setup



GitOps setup:

* Develop: for IT department: deploy, config tests

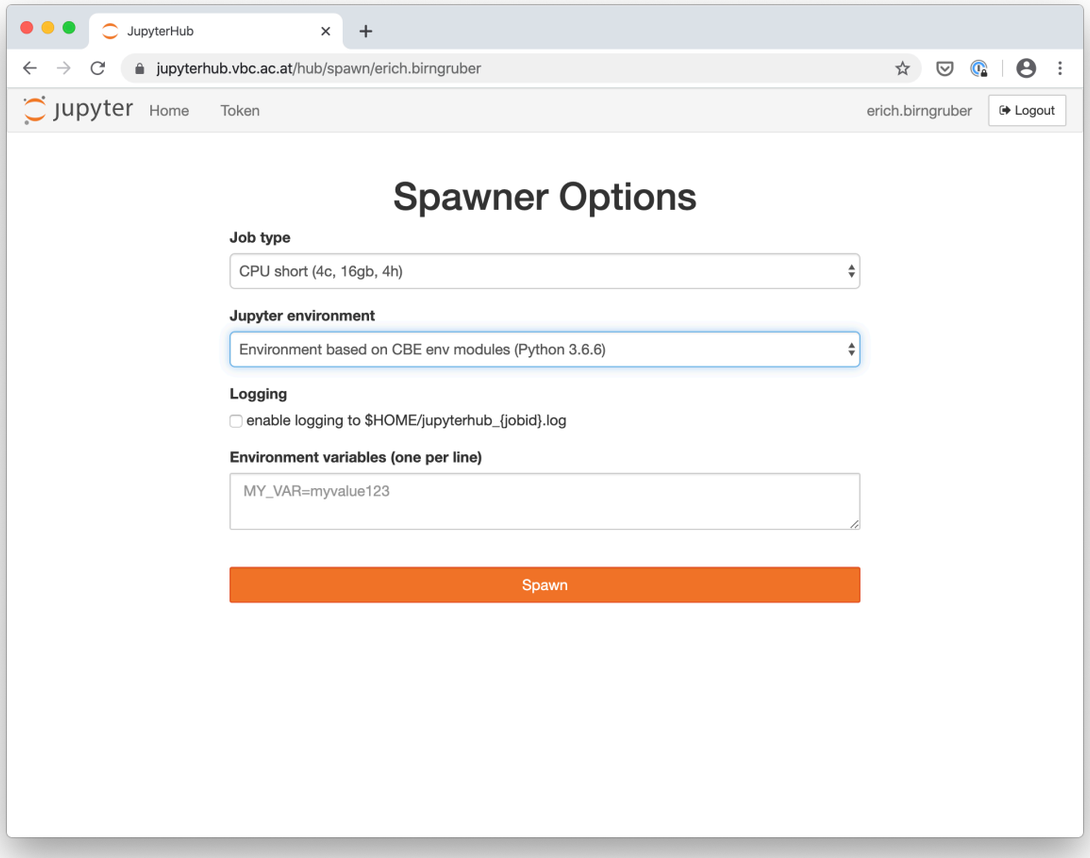
testing: clone of production, for Tool/Workflow developers

production: for end-users

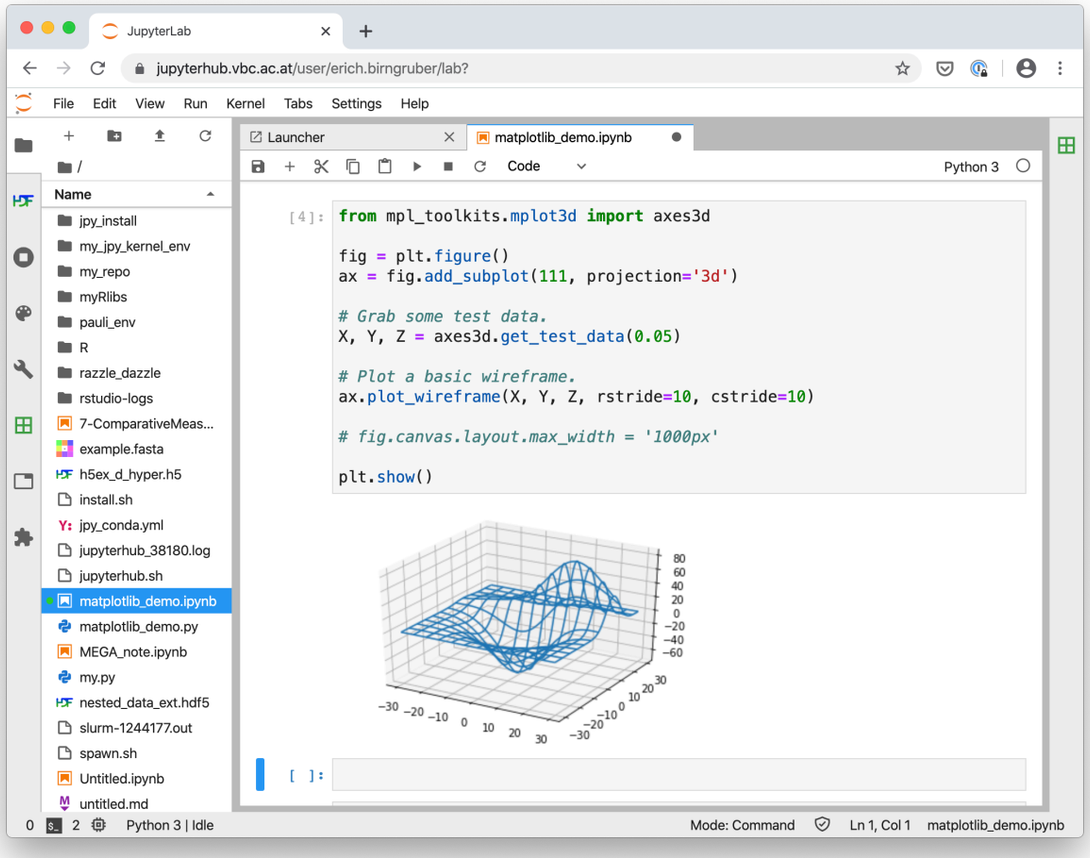
jupyterhub

- https://jupyter.org/
- Web-Based IDE (standalone vs. hub)
- Notebooks = Code + Outputs
- Interpreters as “Kernels”

Notebook: actually JSON
Notebook: Kernel = Interpreter, 1 Kernel per Notebook
Hub: web-connector to individual notebook servers
Hub: allows multiple sessions
Hub: spawners launch



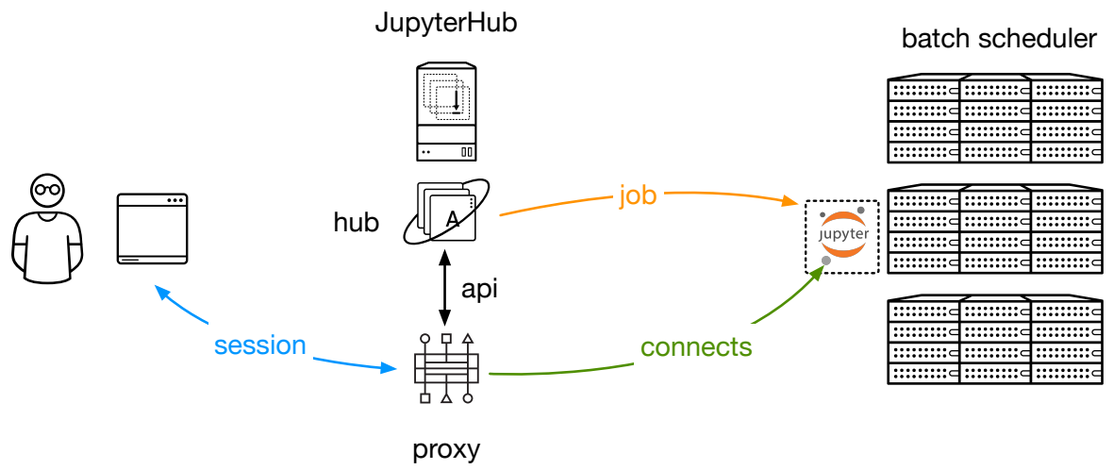
* Spawner = implementation for launching IDE (local, docker, Kubernetes, Batch)
* select job size
* Select environment
    * I.e. use same Python versions (modules) that are available on the cluster
    * maybe run the same code later as patch job
    * There are converters for Notebook -> python script



Jupiter Lab = extended IDE

* File browser
* Notebooks
    * Cells = code snippets, execution unit
* different Kernels
* Various plugins: i.e. viewer for hdf5, FASTA, etc.
* Drawback: no code select -> job like RStudio

JupyterHub setup



Browser connects to hub through a proxy
Hub will program proxy to forward users to their notebook servers
No direct connection to system running the notebooks required

Summary

- XPRA
  Special use cases: X11 applications (Fiji) in Containers

- RStudio
  R (from env modules), web-based IDE

- Galaxy
  pre-configured workflows

- JupyterHub
  Python (per-user kernels), plugins

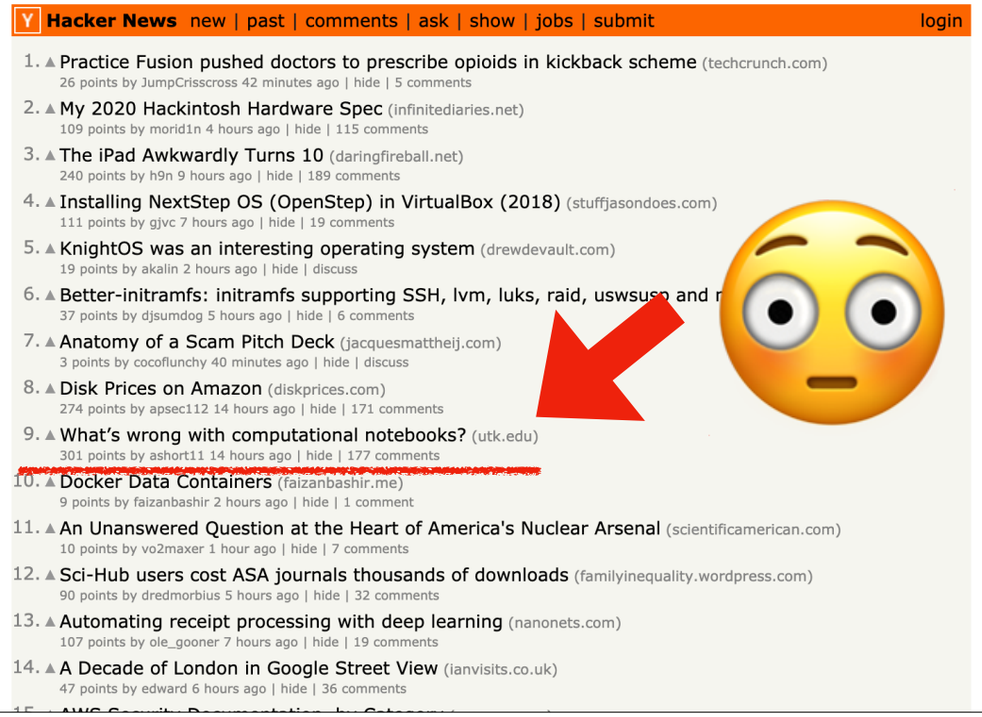
Summary:
XPRA: for special use cases, non-web GUI applications
RStudio: based on module environment, execute code snippets as jobs
Galaxy: workflow tool, UI editor, separate development from production
Jupyterhub: Notebooks, Kernels

Others

- OpenOnDemand: interactive/remote desktop portal
  https://openondemand.org/
- Apache Zeppelin: data exploration “notebooks”
  https://zeppelin.apache.org/
- Eclipse Che: cloud-based editor
  https://www.eclipse.org/che/

This list is non-exhaustive
OpenOnDemand: GUI applications + also web-based shell access - why!
Zeppelin: Datasource (SQL, …) oriented notebooks
Che: cloud-based IDE dev environment - is this where things are moving?

Then this happened



What *is* wrong?

What's Wrong with Computational Notebooks? Pain Points, Needs, and Design Opportunities

Souti Chattopadhyay[1], Ishita Prasad[2], Austin Z. Henley[3], Anita Sarma[1], Titus Barik[2]
Oregon State University[1], Microsoft[2], University of Tennessee-Knoxville[3]
{chattops, anita.sarma}@oregonstate.edu, {ishita.prasad, titus.barik}@microsoft.com, azh@utk.edu

ABSTRACT
Computational notebooks—such as Azure, Databricks, and Jupyter—are a popular, interactive paradigm for data scientists to author code, analyze data, and interleave visualizations, all within a single document. Nevertheless, as data scientists incorporate more of their activities into notebooks, they encounter unexpected difficulties, or pain points, that impact their productivity and disrupt their workflow. Through a systematic, mixed-methods study using semi-structured interviews ($n = 20$) and survey ($n = 156$) with data scientists, we catalog nine pain points when working with notebooks. Our findings suggest that data scientists face numerous pain points throughout the entire workflow—from setting up notebooks to deploying to production—across many notebook environments. Our data scientists report essential notebook requirements, such as supporting data exploration and visualization. The results of our study inform and inspire the design of computational notebooks.

Author Keywords
Computational notebooks; challenges; data science; interviews; pain points; survey

CCS Concepts

Azure,[1] Databricks,[2] Colab,[3] Jupyter,[4] and nteract.[5] While originally intended for exploring and constructing computational narratives [29, 31], data scientists are now increasingly orchestrating more of their activities within this paradigm [33]: through long-running statistical models, transforming data at scale, collaborating with others, and executing notebooks directly in production pipelines. But as data scientists try to do so, they encounter unexpected difficulties—pain points—from limitations in affordances and features in the notebooks, which impact their productivity and disrupt their workflow.

To investigate the pain points and needs of data scientists who work in computational notebooks, across multiple notebook environments, we conducted a systematic mixed-method study using field observations, semi-structured interviews, and a confirmation survey with data science practitioners. While prior work has studied specific facets of difficulties in notebooks [24, 17], such as versioning [18, 19] or cleaning unused code [13, 34], the central contribution of this paper is a taxonomy of validated pain points across data scientists' notebook activities.

Our findings identify that data scientists face considerable pain points through the entire analytics workflow—from set-

20 interviews + 120 surveys
9 Major deficiencies of notebooks
* sharing is "difficult"
* Reproducibility is difficult as it depends on the environment
* Code management:
    * Notebook == JSON
    * Code + data -> changes on every execution
    * Git :(no meaningful diffs

… so as a conclusion: things are changing, different platforms - but with their own problems

References

- XPRA https://xpra.org/
- RStudio https://rstudio.com/
- Jupyterhub https://jupyter.org/hub
- Galaxy https://galaxyproject.org/
- What is wrong with computational notebooks?
  http://web.eecs.utk.edu/~azh/blog/notebookpainpoints.html