

Additional useful tools for HPC

Jupyter lab, Singularity, Snakemake

E. A. Tsatsorin^{1 2}

¹Chief Software developer at CDISE, Skoltech

²Applied Mathematics and Computer Science, MSU

September 2020

Table of Contents

1 Introduction

2 JupyterLab

3 Singularity

4 Snakemake

5 Run on cluster

6 Conclusion

Table of Contents

1 Introduction

2 JupyterLab

3 Singularity

4 Snakemake

5 Run on cluster

6 Conclusion

Introduction

"The Army Field Manual of the Russian Federation"

A soldier is obliged:

- constantly master military professional knowledge, improve their training and military skills
- know and keep in constant readiness for use the weapons and military equipment entrusted to him ...

Principles:

- constantly master professional knowledge
- improve skills
- know equipment



Main Goals of this material

After acknowledge and practice of this material you will be able to:

Main Goals of this material

After acknowledge and practice of this material you will be able to:

- control execution environment even if you don't have root access

Main Goals of this material

After acknowledge and practice of this material you will be able to:

- control execution environment even if you don't have root access
- understand the notion of the data processing pipelines

Main Goals of this material

After acknowledge and practice of this material you will be able to:

- control execution environment even if you don't have root access
- understand the notion of the data processing pipelines
- build reproducible solutions for a variety of tasks involving data processing pipelines

Main Goals of this material

After acknowledge and practice of this material you will be able to:

- control execution environment even if you don't have root access
- understand the notion of the data processing pipelines
- build reproducible solutions for a variety of tasks involving data processing pipelines
- run pipelines locally on PC and on HPC cluster without code modification

Repository

https://github.com/eugtsa/base_singularity

The screenshot shows the GitHub repository page for `eugtsa/base_singularity`. The page includes a navigation bar with links for Pull requests, Issues, Marketplace, and Explore. Below the navigation is a search bar and a star count of 0. The main content area displays a list of files and their commit history:

File	Description	Commit Time
<code>Evgenii Tsatsorin fixed time and memory limits</code>	bigger example first commit	2 hours ago
<code>data</code>	renamed files for demo	4 hours ago
<code>notebooks/ipywidgets</code>	fixed time and memory limits	6 days ago
<code>scripts/snakefile</code>	fixed main README	2 hours ago
<code>README.md</code>	added name to downloaded image	3 hours ago
<code>Singularity</code>	added cmd tools	yesterday
<code>prepare.sh</code>	added name to downloaded image	4 hours ago
<code>requirements.txt</code>	added bigramm lib to requirements	5 days ago

Below the file list is a preview of the `README.md` file, which contains the question "What is this?". The repository has no releases or packages published. The Languages section shows a distribution of code: Jupyter Notebook (99.7%), Python (0.2%), and Shell (0.1%).

Table of Contents

1 Introduction

2 JupyterLab

3 Singularity

4 Snakemake

5 Run on cluster

6 Conclusion

JupyterLab

"The next-generation web-based user interface for Project Jupyter"

The screenshot shows the JupyterLab interface. On the left is a sidebar with tabs for Files, Running, Commands, Cell Tools, and Tabs. Under 'Running', 'Lorenz.ipynb' is selected. In the main area, there are four tabs: 'Lorenz.ipynb' (active), 'Terminal 1', 'Console 1', and 'Data.ipynb'. The 'Lorenz.ipynb' tab contains text about the Lorenz system and three differential equations:

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

Below the equations, it says: "Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors." A code cell in 'In [4]:' shows:

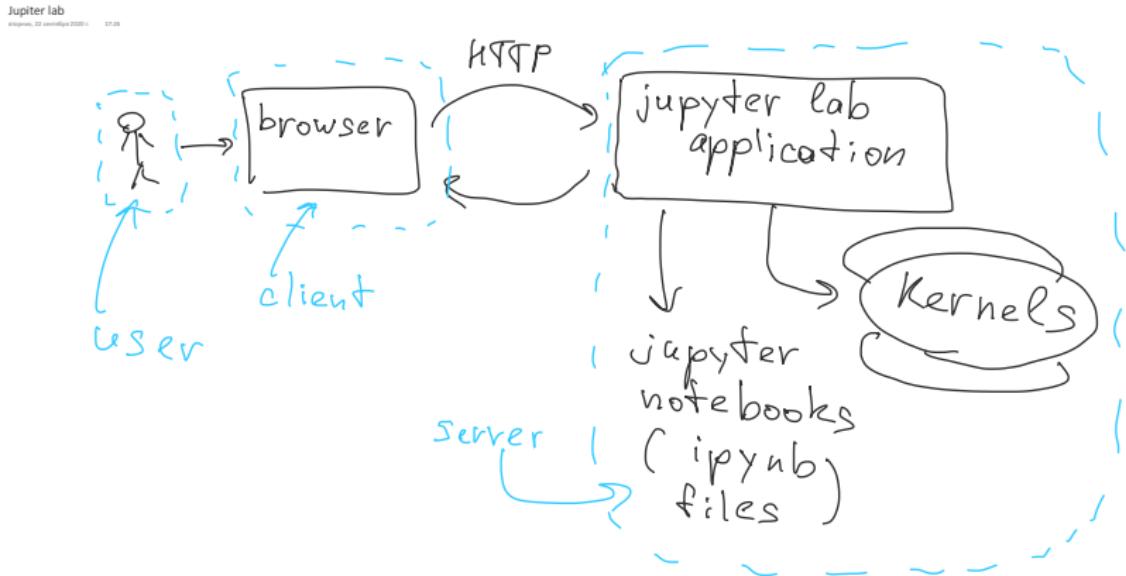
```
from lorenz import solve_lorenz
t, x, _ = solve_lorenz(N=10)
```

Below the code cell is an 'Output View' panel with sliders for 'sigma' (10.00), 'beta' (2.67), and 'rho' (28.00). To the right is a 3D plot of the Lorenz attractor, showing a complex, swirling trajectory. At the bottom of the screen, there are navigation icons for back, forward, search, and other functions.

SkoTech
Moscow Institute of Science and Technology

MSU
Moscow State University

JupyterLab architecture



JupyterLab





It's like Jupyter Notebook, but better. Why?

JupyterLab

Syntax highlight, Autocompletion

The screenshot shows the JupyterLab interface running in Mozilla Firefox. On the left, there is a file browser window displaying a list of files and folders. On the right, there is a code editor window titled "utils.py" containing Python code. The code editor has syntax highlighting and autocompletion features.

```
for indicator_index in self._indicator_indexes:
    current_len = 0
    current_position = indicator_index - 1

    while y[current_position] == 1:
        current_len += 1
        current_position -= 1

    self._y_lengths.append(current_len)
    self._add_first_lengths = add_first_lengths
    self._add_last_lengths = add_last_lengths
    self._y = y.copy()
    self._binary_size = (self._n_intervals + len(self._add_first_lengths) +
len(self._add_last_lengths)) * len(
        self._indicator_indexes)

def get_binary_size(self):
    """
    Get size of binary vector needed for y creation
    :return: int, size of binary vector needed for y creation
    """
    return self._binary_size

def _split_for(self, len_to_split, n):
    to_split = len_to_split
    while to_split != 0:
        cur_result = math.ceil(to_split / n)
        yield cur_result
        n -= 1
        to_split = to_split - cur_result

def get_y_by_mask(self, bimask):
    """
```

Skoltech
Moscow Institute of Science and Technology



JupyterLab

Understands many formats: JSON

The screenshot shows a JupyterLab interface running in Mozilla Firefox. On the left, there is a file tree view showing various files and folders. In the center, there is a code editor window titled "development.json" containing JSON code. The JSON code defines a configuration for a metrics service.

```
root: {} 4 keys
  config.name: "bio_db_metrics"
  config.version: "0.1"
  * metrics_service_config: {} 9 keys
    port: 27017
    host: "10.30.16.181"
    db: "bio_db"
    * tasks_to_use_metrics: [] 2 items
      0: "back_hurt_simple_v1"
      1: "back_hurt_simple_v3"
    * authors_to_check: [] 2 items
      0: "Eugene.Tsatsorin"
      1: "Aleksandr.Medvedev"
    hashed_fields: []
    rebuild_cache: true
    update_interval: 2
    print_progress: false
  * logging: {} 3 keys
```

JupyterLab

Understands many formats: Markdown

The screenshot shows the JupyterLab interface running in Mozilla Firefox. On the left, a file tree displays a directory structure under 'example'. In the main area, a code editor shows a Markdown file named 'Markdown_demo.md' containing the following content:

```
h1 Heading 8-)
h2 Heading
h3 Heading
h4 Heading
h5 Heading
h6 Heading

Horizontal Rules


---




---



## Typographic replacements



Enable typographer option to see result.



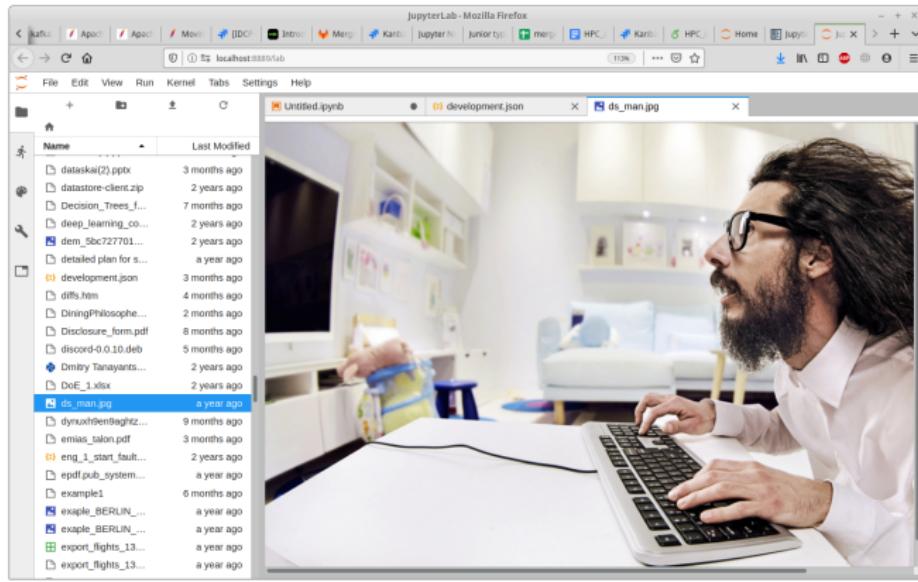
(c) (C) (r) (R) (tm) (TM) (p) (P) +  

test... test.... test?.... test!....


```

JupyterLab

Understands many formats: Images



Skoltech
Moscow Institute of Science and Technology



JupyterLab

Understands many formats: CSV, TSV

Browser	Max Size
Firefox	1.04GB
Chrome	730MB
Safari	1.8GB

The screenshot shows the JupyterLab interface running in Mozilla Firefox. At the top, there's a navigation bar with links like Home, Help, and Settings. Below it is a toolbar with icons for File, Edit, View, Run, Kernel, Tabs, and Settings. A central area contains a file browser and a terminal window.

File Browser: On the left, a tree view shows a directory structure. One folder, 'merged_tf_simple', is expanded, showing its contents. The files listed include various CSV and TSV files, along with other data files like 'kpi_avg_each_cell_for_all_days.xls' and 'metrics_provider.py'. There are also several PDF files and a 'memoch_142_ch1_1.pdf' document.

Terminal: On the right, a terminal window titled 'merged_tf_simple.csv' is open. It displays a table with 28 rows and 4 columns. The columns are labeled 'Name', 'Last modified', 'Size', and 'Preview'. The data includes file names like 'kpi_avg_each_cell_for_all_days.xls' and their corresponding details such as size (e.g., 18.0 MB) and preview snippets.

Skoltech

Moscow Institute of Science and Technology



JupyterLab

Works with extensions, for example Ipywidgets:

<https://ipywidgets.readthedocs.io/en/latest/index.html>

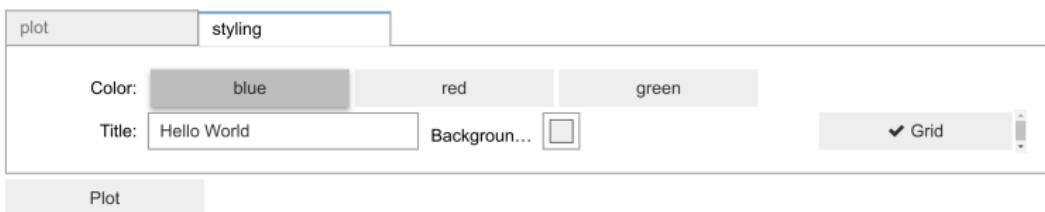
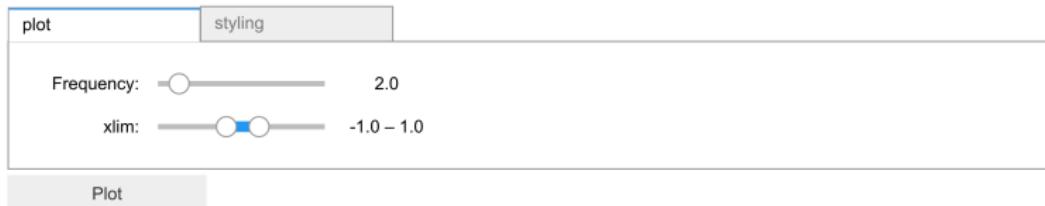


Table of Contents

1 Introduction

2 JupyterLab

3 Singularity

4 Snakemake

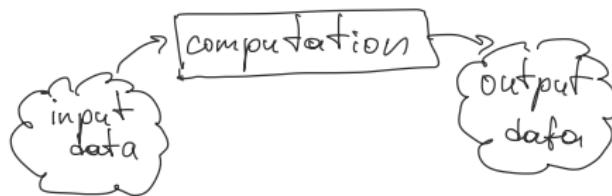
5 Run on cluster

6 Conclusion

Abstract computation

Naive notion of computation

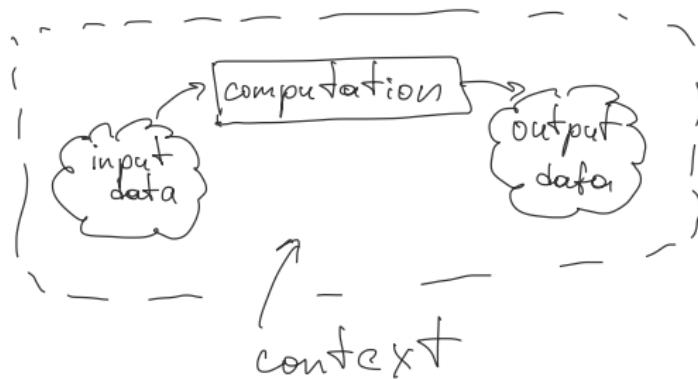
Simple computation
sprint, 23 seconds ago • 10:01



Abstract computation

Less naive notion of computation

Simple computation
smpc, 19 seconds 2021-11-13 10



Singularity

Free, cross-platform and open-source containerization engine

Singularity is the Container of Choice for Performance Critical Applications

Singularity is currently used by academia, government agencies, national labs, biotechnology, oil and gas, aerospace, EDA, and many other industries.



Stanford
University

HARVARD
UNIVERSITY



U.S. DEPARTMENT OF
ENERGY



SDSC
SAN DIEGO
SUPERCOMPUTER CENTER

Skoltech
Moscow Institute of Science and Technology



Containerization

Containerization is defined as a form of operating system virtualization, through which applications are run in isolated user spaces called containers, all using the same shared operating system (OS).



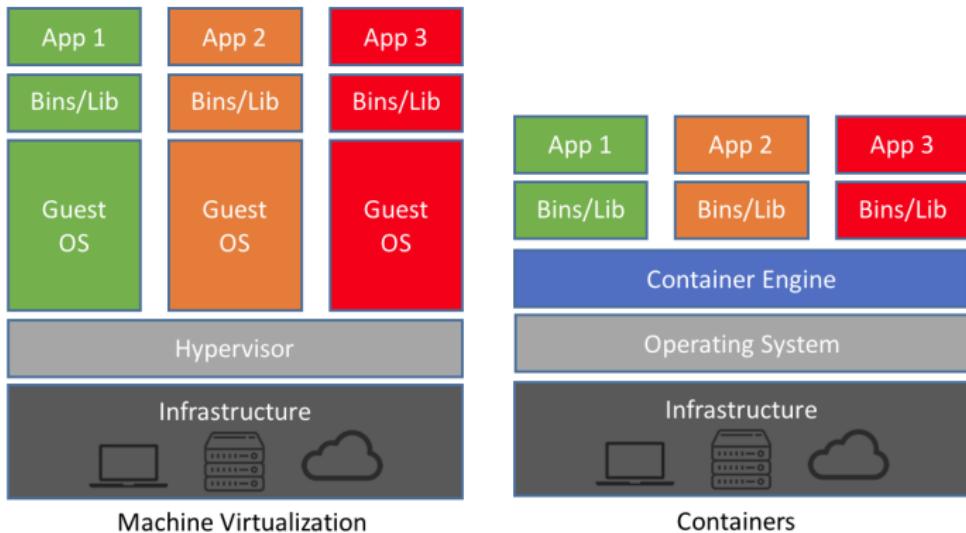
Containerization

Containerization is defined as a form of operating system virtualization, through which applications are run in isolated user spaces called containers, all using the same shared operating system (OS).



It's a tool to preserve dependencies!

Containerization vs Virtualization



Why singularity?

Why singularity is very suitable for scientific computing and HPC?

- Runs without sudo or special groups - great for security

Why singularity?

Why singularity is very suitable for scientific computing and HPC?

- Runs without sudo or special groups - great for security
- Runs container processes without a daemon - great for scheduling

Why singularity?

Why singularity is very suitable for scientific computing and HPC?

- Runs without sudo or special groups - great for security
- Runs container processes without a daemon - great for scheduling
- Native conversion from docker image to singularity - great for reusing docker images

Why singularity?

Why singularity is very suitable for scientific computing and HPC?

- Runs without sudo or special groups - great for security
- Runs container processes without a daemon - great for scheduling
- Native conversion from docker image to singularity - great for reusing docker images
- Created for human users, not for services - great for manual commands

Created for humans

To run with nvidia cuda support use '--nv' flag:

```
$ singularity exec --nv base_singularity.sif jupyter lab
```

```
[base] tsatsorin.home:~/singl$ singularity exec --nv ./base_singularity/base.sing jupyter lab --port 19999
[1;23:13:12.727 LabApp] JupyterLab extension loaded from /usr/local/lib/python3.7/site-packages/jupyterlab
[1;23:13:12.728 LabApp] JupyterLab application directory is /usr/local/share/jupyter/lab
[W 23:13:12.728 LabApp] JupyterLab extension not enabled, manually loading...
[1;23:13:12.729 LabApp] JupyterLab extension loaded from /usr/local/lib/python3.7/site-packages/jupyterlab
[1;23:13:12.730 LabApp] JupyterLab extension not enabled, manually loading...
[1;23:13:12.730 LabApp] Setting notebook_dir from local directory: /usr/local/share/jupyter/lab
[1;23:13:12.730 LabApp] The Jupyter Notebook is running at:
[1;23:13:12.730 LabApp] http://localhost:19999/?token=9aa7fe2a2ab7affc49a1983f85be9e4df7cad429ba96a
[1;23:13:12.730 LabApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 23:13:12.734 LabApp] No web browser found: could not locate runnable browser.
[C 23:13:12.734 LabApp]

To access the notebook, open this file in a browser:
file:///home/tsat/local/share/jupyter/runtime/nbsviewer-1560-open.html
Or copy and paste one of these URLs:
http://localhost:19999/?token=9aa7fe2a2ab7affc49a1983f85be9e4df7cad429ba96a
[1;23:13:20.050 LabApp] 362 GET /?token=9aa7fe2a2ab7affc49a1983f85be9e4df7cad429ba96a (127.0.0.1) 1.38ms
[1;23:13:25.735 LabApp] Node v12.4.0

[1;23:13:25.736 LabApp] Build is up to date
[1;23:13:34.762 LabApp] Kernel started: 6cc4dc03 4f87 4f29 9c54 762a927e3d78
[1;23:13:34.023 LabApp] Adapting to protocol v5.1 for kernel 6cc4dc03-4f87-4f29-9c54-762a927e3d78
[1;23:13:35.047 LabApp] Adapting to protocol v5.1 for kernel 6cc4dc03-4f87-4f29-9c54-762a927e3d78
[1;23:14:01.888 LabApp] Kernel interrupted: 6cc4dc03-4f87-4f29-9c54-762a927e3d78
[1;23:14:03.948 LabApp] Kernel interrupted: 6cc4dc03-4f87-4f29-9c54-762a927e3d78
[1;23:15:34.205 LabApp] Saving file at /notebooks/from_kaggle/introduction-to-sound-event-detection.ipynb
[1;23:17:34.627 LabApp] Saving file at /notebooks/from_kaggle/introduction-to-sound-event-detection.ipynb
[[{"A": "1"}, {"A": "1"}, {"A": "1"}, {"A": "1"}, {"A": "1"}]]
```

```
KubePod2,RC: nvidia-smi                               tsa-home: Mon Sep 23 23:19:29 2020
Wed Sep 23 23:19:29 2020
+-----+
| NVIDIA-SMI 450.57   Driver Version: 450.57    CUDA Version: 11.0 |
+-----+
| GPU  Name        Persistence-M  Bus-Id      Disp.A  Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr-Usage/Cap | Memory-Usage | GPU-Util  Compute M. |
|          %          %          %          %          %          %          %          %          |
+-----+
|  0  GeForce GTX 106... Off  00000000:01:00.0  On           N/A     Default |
| 61% 75C  P2  106M / 120M  4568MiB /  6070MiB   99%  Default |
|          |          |          |          |          |          |          |          |
+-----+
Processes:
GPU ID CI PID Type Process name          GPU Memory Usage
+-----+
| 0  N/A  N/A  3659  C  /usr/local/bin/python  4457MiB
| 0  N/A  N/A  2024  G  /usr/lib/xorg/Xorg  1860MiB
+-----+
```



Skoltech
National Institute of Science and Technology

Singularity Hub

Use singularity-hub.org:

- native integration with github
- free worker to build and update your image
- easy to pull container from internet



Singularity Container Registry

A collaboration between [Stanford University](#) and [SingularityLLC](#)

Singularity Hub is open for general use. You will need to update webhooks to remove the leading www., or open an issue to get help. Please see the [Release Notes](#) for important details about new usage limits.

Skoltech
Moscow Institute of Science and Technology



Main singularity commands

Main singularity commands:

- build: `$ sudo singularity build [your_image_file] Singularity`
- pull image: `$ singularity pull shub:[your_image_link]`
- run image: `$ singularity exec [your_image_file] [command_to_run]`

Singularity example

clone: \$ git clone https://github.com/eugtsa/base_singularity.git
build: \$ sudo singularity build base.simg Singularity
run image: \$ singularity run base_singularity.simg jupyter lab

```
1  Bootstrap: docker
2  From: neurodebian:latest
3
4  %help
5
6      Container with packages on top of base 3.6 with anaconda
7
8  %files
9      ./requirements.txt /requirements.txt
10
11 %post
12
13     apt-get update
14     DEBIAN_FRONTEND=noninteractive apt-get -yq install \
15         build-essential \
16         wget \
17         tmux \
18         tree \
19         vim \
20         git \
21         libsoffice1 \
22         postgresql-server-dev
23
24     rm -rf /var/lib/apt/lists/*
25     apt-get clean
26
27     wget -c https://repo.anaconda.com/archive/Anaconda3-2019.03-Linux-x86_64.sh
28     /bin/bash Anaconda3-2019.03-Linux-x86_64.sh -bfp /usr/local
29
30     #Conda configuration of channels from .condarc file
31
32     conda config --add channels defaults
33     conda config --add channels conda-forge
34     conda config --add channels pytorch
35     conda config --add channels mepes
36     conda update conda
37     pip install --upgrade pip
38     rm -rf /usr/local/lib/python3/site-packages/livelibelite*
39
40     #Install environment
41     rm -rf ~/anaconda3/lib/python3.6/site-packages/livelibelite*
42     pip install -I -r requirements.txt
43     conda install nodejs
44     jupyter labextension install @jupyter-widgets/jupyterlab-manager
```

Table of Contents

1 Introduction

2 JupyterLab

3 Singularity

4 Snakemake

5 Run on cluster

6 Conclusion

Snakemake

The Snakemake workflow management system is a tool to create reproducible and scalable data analyses.

The screenshot shows the official Snakemake documentation website. At the top left is the Snakemake logo, which consists of a stylized green 'S' icon followed by the word 'snakemake' in a lowercase sans-serif font. Below the logo is the word 'stable'. A search bar with the placeholder 'Search docs' is located below the logo. On the left side, there are two sections of links: 'GETTING STARTED' and 'EXECUTING WORKFLOWS'. The 'GETTING STARTED' section includes links for 'Installation', 'Snakemake Tutorial', 'Short tutorial', and 'Snakemake Executor Tutorials'. The 'EXECUTING WORKFLOWS' section includes links for 'Command line interface', 'Cluster Execution', 'Cloud execution', 'Between workflow caching', and 'Interoperability'.

[Docs](#) » Snakemake

[Edit on GitHub](#)

Snakemake

239k python 3.5 pypi v5.24.1 docker container passing CI passing

stack overflow Follow 1.7k discord chat 3 online Stars 650

The Snakemake workflow management system is a tool to create reproducible and scalable data analyses. Workflows are described via a human readable, Python based language. They can be seamlessly scaled to server, cluster, grid and cloud environments, without the need to modify the workflow definition. Finally, Snakemake workflows can entail a description of required software, which will be automatically deployed to any execution environment.

Snakemake is highly popular with, >5 new citations per week. For an introduction, please visit <https://snakemake.github.io>.



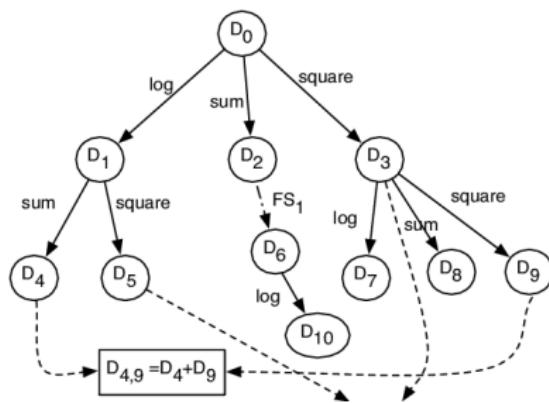
Problem

You need to write a computation involving some data, and:

- data doesn't fit into PC RAM
- you can parallelize steps of computation slicing it into independent chunks
- you can tie steps together through inputs and outputs

Pipelines

Pipeline is a set of data processing elements connected in series, where the output of one element is the input of the next one. Pipeline is represented as DAG. The elements of a pipeline are often executed in parallel or in time-sliced fashion.



Pipelines in snakemake

Pipelines in Snakemake are:

- inspired by make utility
- built in DSL on top of python
- planned in bottom-up, executed in top-bottom manner

Tell Snakemake what files you want to be created

```
rule:  
    input: "A.txt", "B.txt", "C.txt"
```

Produce the files you want to have from some intermediate result

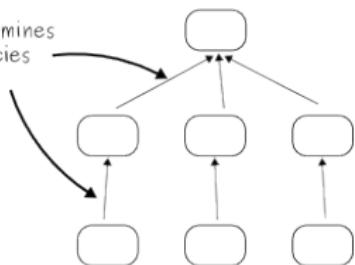
```
rule:  
    input: "{sample}.inter"  
    output: "{sample}.txt"  
    shell: "somecommand {input} {output}"
```

Create a needed intermediate result

```
rule:  
    input: "{sample}.in"  
    output: "{sample}.inter"  
    run:  
        somepythoncode()
```

Snakemake determines the dependencies for you

Use wildcards to write general rules for all samples



Rule

Pipelines in Snakemake are built from rules

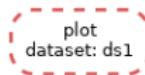
```
rule plot:  
    output: "{dataset}_plot.jpg"  
    input:  "{dataset}.csv"  
    shell: "python plotter.py {input} {output}"
```

- rule consists of input, output and shell or run command
- rule is generalized by wildcards

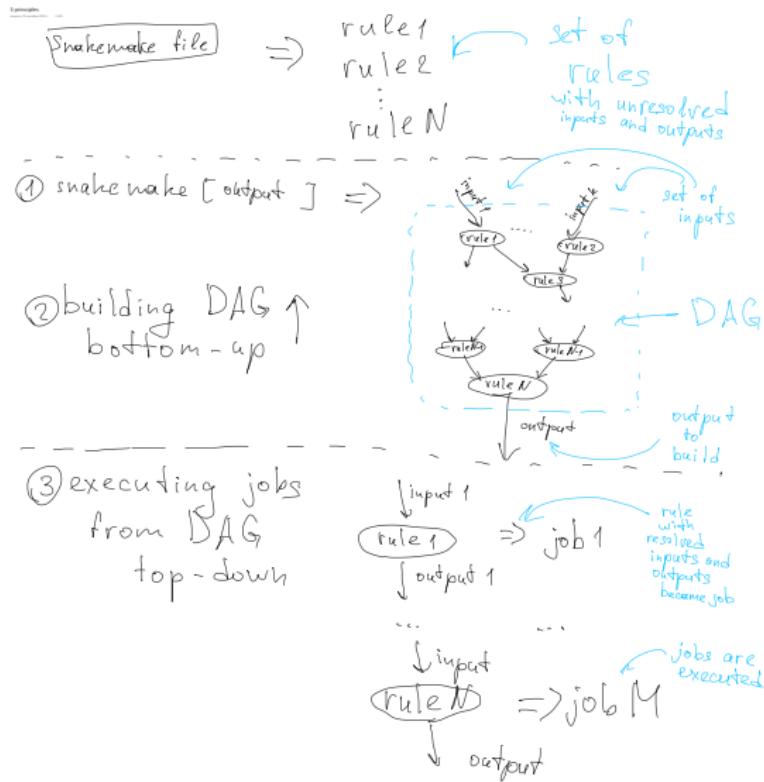
```
$ cd ./scripts/snakefile/1_tutorial  
$ snakemake ds1_plot.jpg
```

```
(base) tsa@tsa-T480S:~/git_work2/base_singularity/scripts/snakefile/1_tutorial$ snakemake ds1_plot.jpg --cores all  
Building DAG of jobs...  
Using shell: /bin/bash  
Provided cores: 8  
Rules claiming more threads will be scaled down.  
Job counts:  
    count   jobs  
        1     plot  
        1  
[Tue Sep 22 19:44:55 2020]  
rule plot:  
    input: ds1.csv  
    output: ds1.plot.jpg  
    jobid: 0  
    wildcards: dataset=ds1  
Attribute Qt::AA_EnableHighDpiScaling must be set before QCoreApplication is created.  
[Tue Sep 22 19:44:56 2020]  
finished job 0.  
1 of 1 steps (100%) done  
Complete log: /home/tsa/git_work2/base_singularity/scripts/snakefile/1_tutorial/.snakemake/log/2020-09-22T194455.349576.snake
```

```
$ snakemake --dag ds1_plot.jpg --cores all | dot -Tpng > dag.png
```



Pipelines in snakemake



\$ snakemake ds1_filtered_plot.jpg

```
(base) tsa@tsa-T480S:~/git/work2/base_singularity/scripts/snakefile/tutorials/1_tutorial$ snakemake ds1_filtered_plot.jpg --cores all
Building DAG of jobs...
Using shell: /bin/bash
Provided cores: 8
Rules claiming more threads will be scaled down.
Job counts:
    count   jobs
        1     filter
        1     plot
        2

[Tue Sep 22 20:03:50 2020]
rule filter:
    input: ds1.csv
    output: ds1_filtered.csv
    jobid: 1
    wildcards: csvdata=ds1

[Tue Sep 22 20:03:50 2020]
Finished job 1.
1 of 2 steps (50%) done

[Tue Sep 22 20:03:50 2020]
rule plot:
    input: ds1_filtered.csv
    output: ds1_filtered_plot.jpg
    jobid: 0
    wildcards: dataset=ds1_filtered

Attribute Qt::AA_EnableHighDpiScaling must be set before QCoreApplication is created.
[Tue Sep 22 20:03:51 2020]
Finished job 0.
2 of 2 steps (100%) done
Complete log: /home/tsa/git/work2/base_singularity/scripts/snakefile/tutorials/.snakemake/log/2020-09-22T200350.389003.snakemake.log
```

\$ snakemake --dag ds1_filtered_plot.jpg --cores all | dot -Tpng
> dag2.png

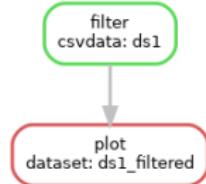


Table of Contents

1 Introduction

2 JupyterLab

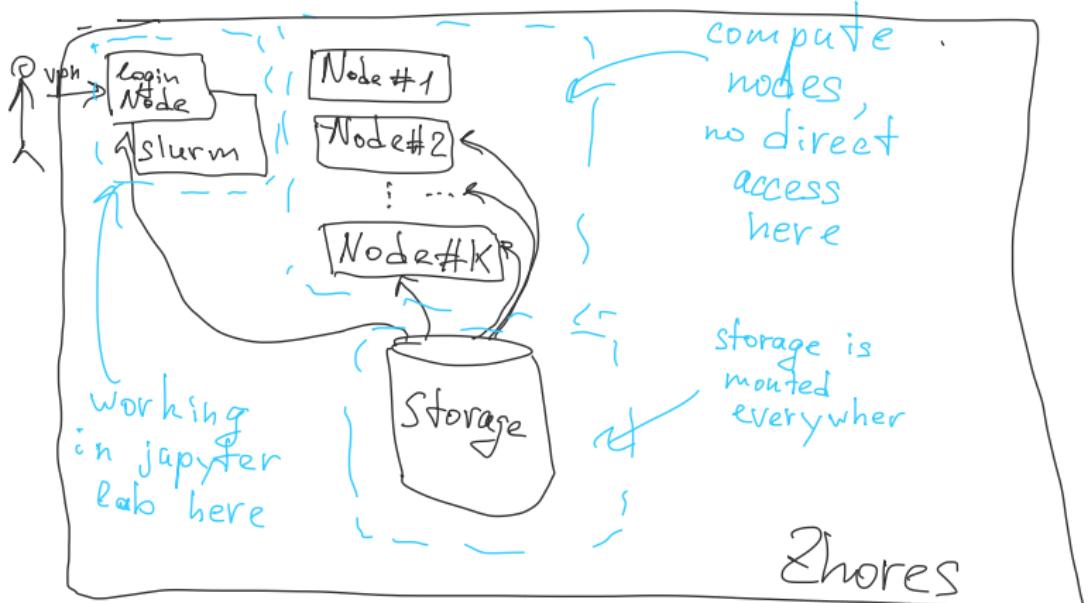
3 Singularity

4 Snakemake

5 Run on cluster

6 Conclusion

Cluster architecture simplified



Slurm

The Slurm ... is a free and open-source job scheduler for Linux and Unix-like kernels.

- allocating exclusive and/or non-exclusive access to resources (computer nodes) to users for some duration of time so they can perform work

The Slurm ... is a free and open-source job scheduler for Linux and Unix-like kernels.

- allocating exclusive and/or non-exclusive access to resources (computer nodes) to users for some duration of time so they can perform work
- providing a framework for starting, executing, and monitoring work

Slurm

The Slurm ... is a free and open-source job scheduler for Linux and Unix-like kernels.

- allocating exclusive and/or non-exclusive access to resources (computer nodes) to users for some duration of time so they can perform work
- providing a framework for starting, executing, and monitoring work
- arbitrating contention for resources by managing a queue of pending jobs

Slurm

The Slurm ... is a free and open-source job scheduler for Linux and Unix-like kernels.

- allocating exclusive and/or non-exclusive access to resources (computer nodes) to users for some duration of time so they can perform work
- providing a framework for starting, executing, and monitoring work
- arbitrating contention for resources by managing a queue of pending jobs

Slurm is the workload manager on about 60% of the TOP500 supercomputers

Snakemake + slurm

In order to run snakemake job you need to create cluster.yml config

```
__default__:  
    account: e.tcatcorin  
    partition: cpu_small  
    time: 00:05:00 # time limit for each job  
    nodes: 1  
    ntasks-per-node: 1 #Request n cores be allocated per node.  
    chdir: /gpfs/data/home/e.tcatcorin/base_singularity/scripts  
        /snakemake/2_tutorial_with_singularity  
    output: snakemake_simple-%j.out  
    error: snakemake_simple-%j.err  
    memory: 500
```

Run on cluster

```
$ cd ./scripts/snakefile/3_bigger_example  
$ snakemake --use-singularity --cluster "sbatch -t cluster.time -p  
cluster.partition -N cluster.nodes --mem cluster.memory" --cluster-config  
cluster_config.yml --jobs 20 proj.vw
```

```
[base] tsat@tsa-T480s:~/git_work2/base_singularity/scripts/snakefile/1_tutorial$ snakemake dsl_plot.jpg --cores all  
Building DAG of jobs...  
Using shell: /bin/bash  
Provided cores: 8  
Rules claiming more threads will be scaled down.  
Job counts:  
 count   jobs  
    1      plot  
    1  
[Tue Sep 22 19:44:55 2020]  
rule plot:  
    input: dsl.csv  
    output: dsl.plot.jpg  
    jobid: 0  
    wildcards: dataset=dsl  
  
Attribute Qt::AA_EnableHighDpiScaling must be set before QCoreApplication is created.  
[Tue Sep 22 19:44:56 2020]  
Finished job 0.  
1 of 1 step(s) (100%) done  
Complete log: /home/tsat/git_work2/base_singularity/scripts/snakefile/1_tutorial/.snakemake/log/2020-09-22T194455_349576_snake
```

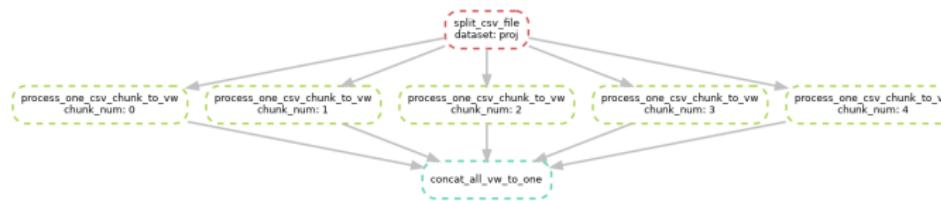


Table of Contents

1 Introduction

2 JupyterLab

3 Singularity

4 Snakemake

5 Run on cluster

6 Conclusion

Now you know the basics of:

- Jupyter lab

Now you know the basics of:

- Jupyter lab
- Singularity containerization

Now you know the basics of:

- Jupyter lab
- Singularity containerization
- Snakemake

Now you know the basics of:

- Jupyter lab
- Singularity containerization
- Snakemake
- Snakemake with singularity

Now you know the basics of:

- Jupyter lab
- Singularity containerization
- Snakemake
- Snakemake with singularity
- How to run it on cluster

https://github.com/eugtsa/base_singularity

In or

eugtsa@gmail.com

e.tcatcorin@skoltech.ru