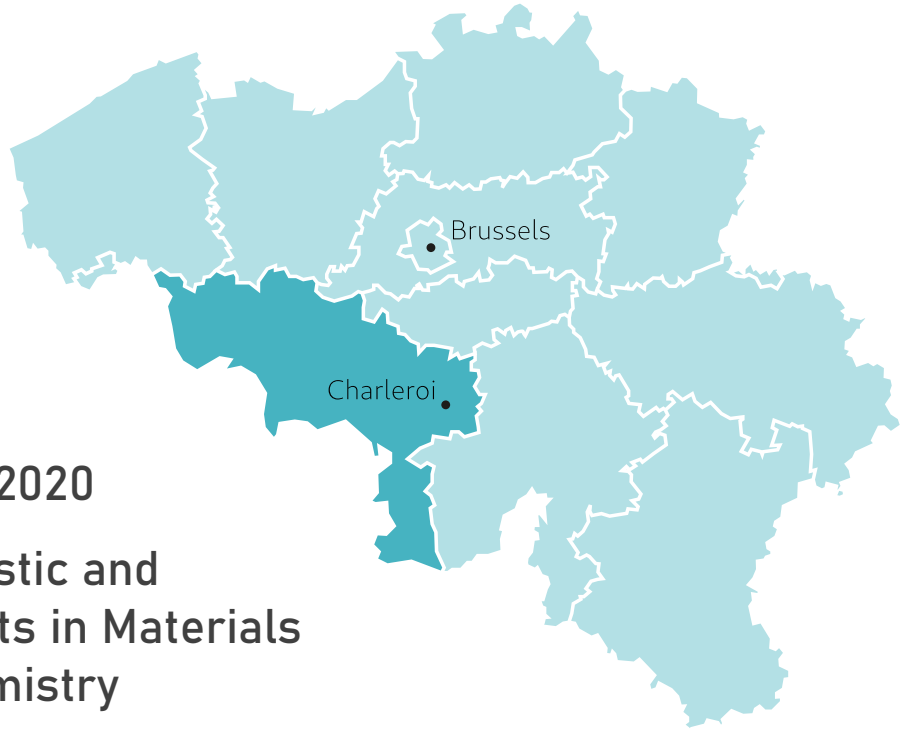
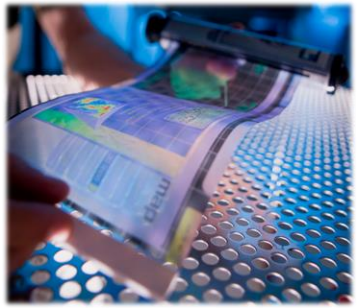
Matgenix

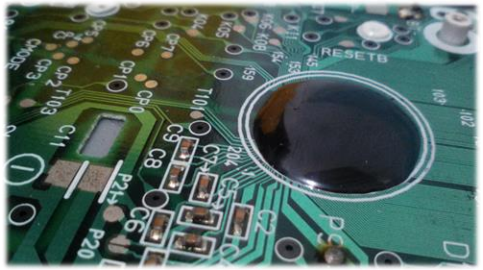
WHO/WHERE WE ARE



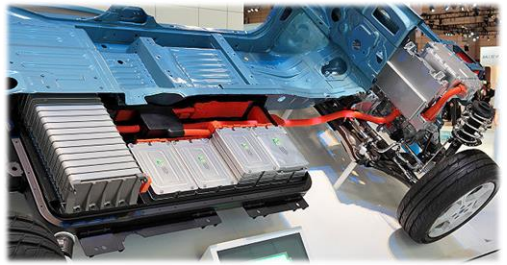
Founded in June 2020

Team of enthusiastic and passionate experts in Materials Science and Chemistry





AS FAR AS TECHNOLOGY IS CONCERNED, MATERIALS AND CHEMICAL ADVANCES ARE NEEDED

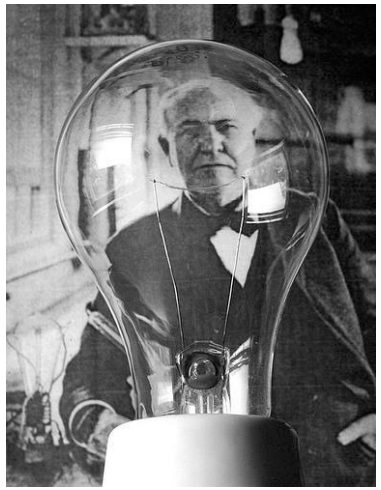


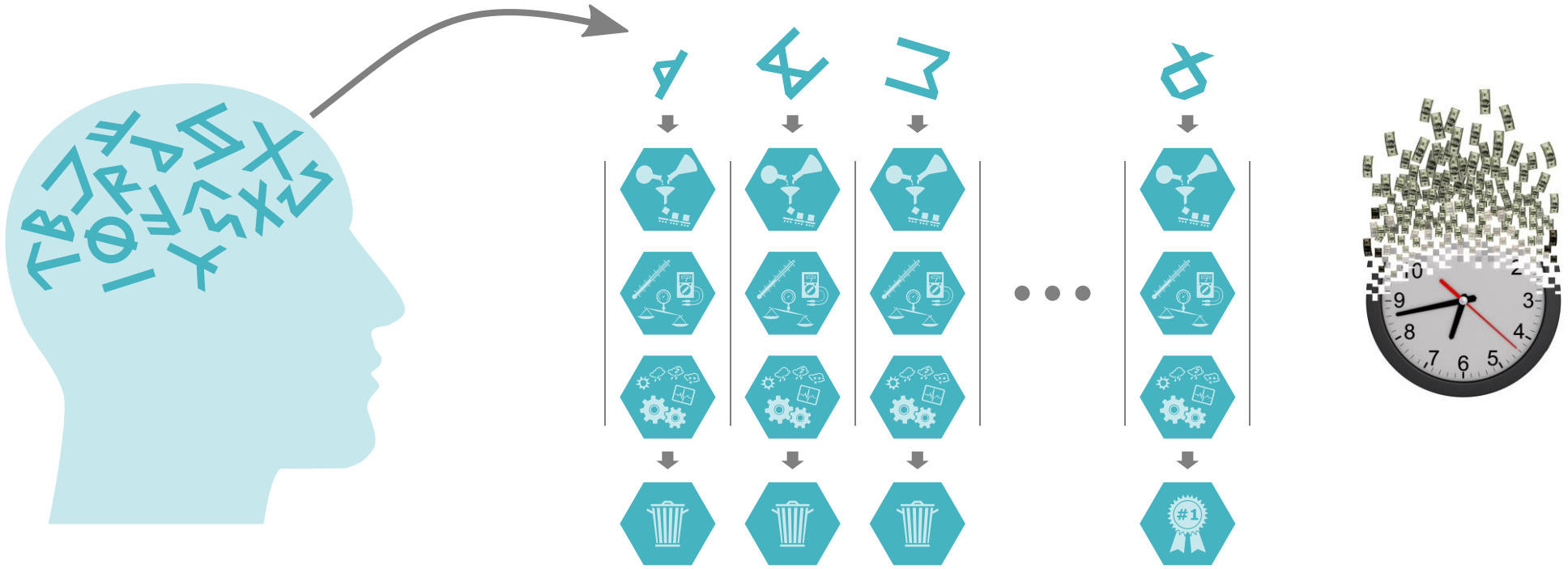
HOW TO FIND THE BEST MATERIALS AND CHEMICALS

HOW TO FIND THE BEST MATERIALS AND CHEMICALS

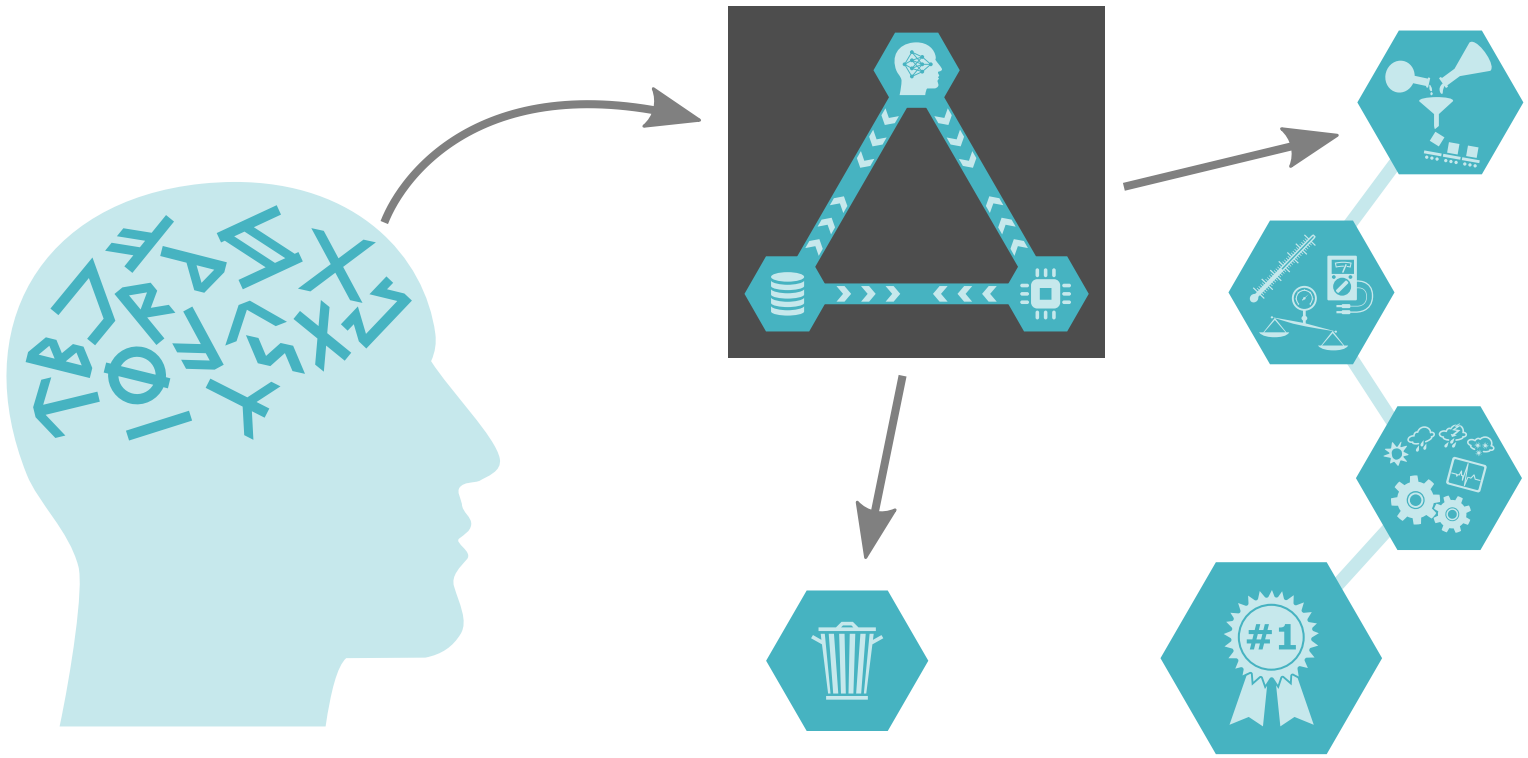
HOW TO FIND THE BEST MATERIALS AND CHEMICALS

MATERIALS DESIGN: THE TRADITIONAL APPROACH





MATERIALS DESIGN: THE IN SILICO APPROACH

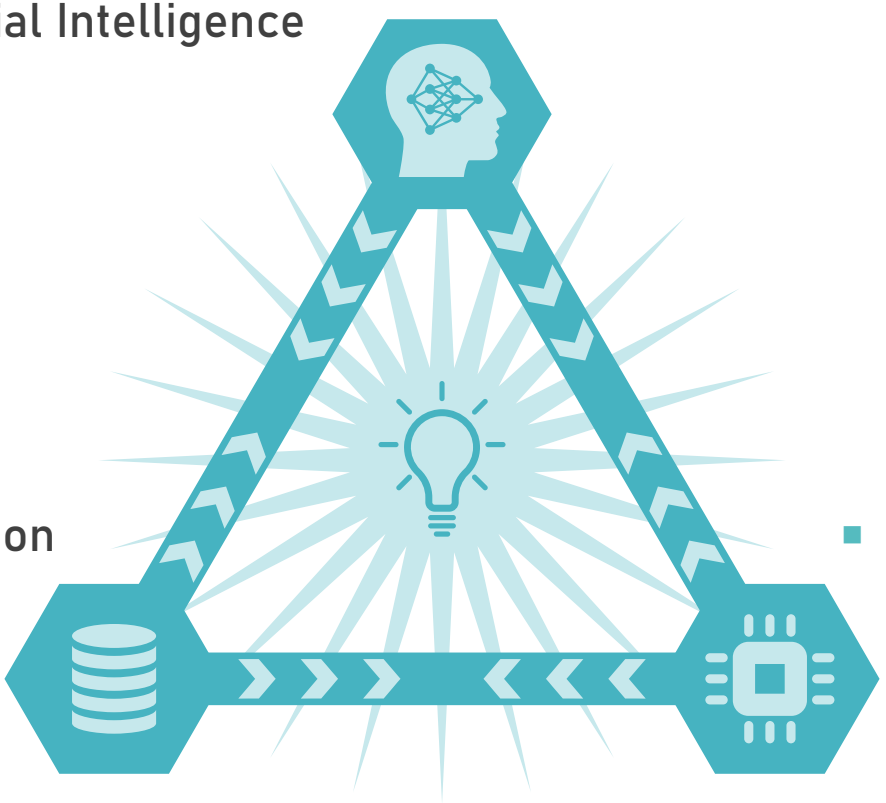


EASIER TO SCALE UP !

And the tools are available!!!

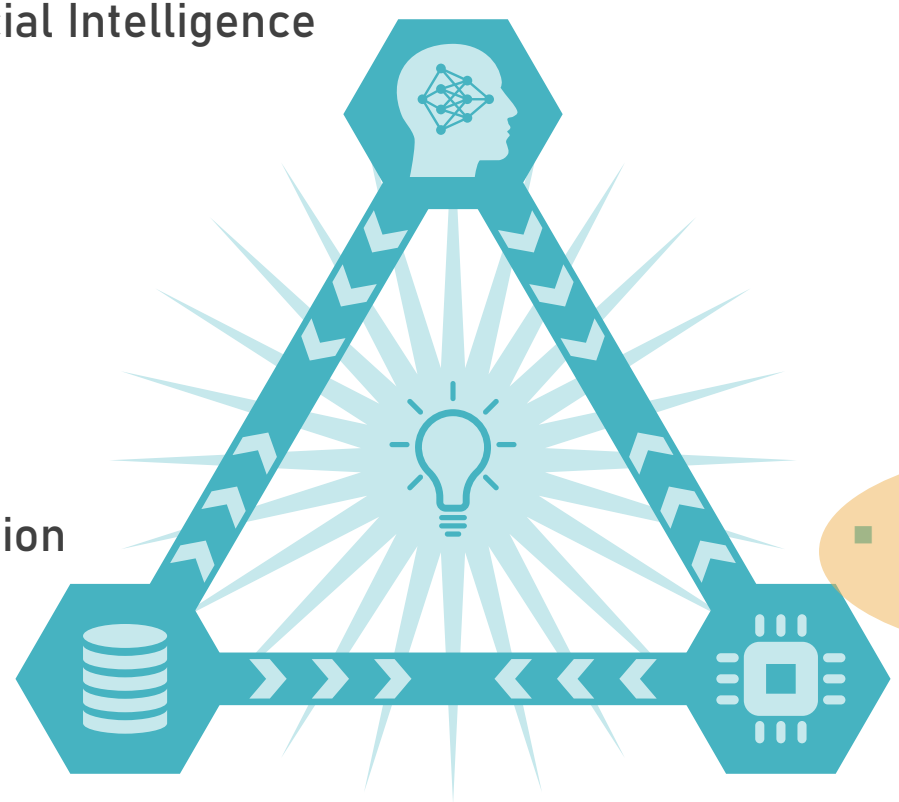
MATERIALS DESIGN: THE THREE PILLARS

- Artificial Intelligence
- Data consolidation
- High-Throughput simulations



MATERIALS DESIGN: THE THREE PILLARS

- Artificial Intelligence



- Data consolidation
- High-Throughput simulations

HIGH-THROUGHPUT SIMULATIONS

ATOMISTIC SIMULATIONS

- When you run few simulations, everything is (more or less) fine

ATOMISTIC SIMULATIONS

- When you run many simulations (high-throughput), things get more complex …

ATOMISTIC SIMULATIONS

- When you run many simulations (high-throughput), things get (much) more complex …

Need automatic tools to deal with high-throughput simulations

HIGH-THROUGHPUT SIMULATIONS

- High volume of calculated properties/molecules/materials through an automatized process (= Workflow)

HIGH-THROUGHPUT SIMULATIONS

- High volume of calculated properties/molecules/materials through an automatized process (= Workflow)
- Several steps involved:

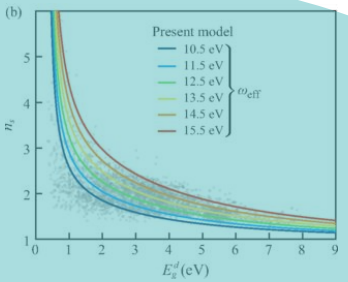
Input generation

Code execution

Error corrections

Parsing outputs

Data storage



Data analysis

HIGH-THROUGHPUT SIMULATIONS

- High volume of calculated properties/molecules/materials through an automatized process (= Workflow)
- Several steps involved:

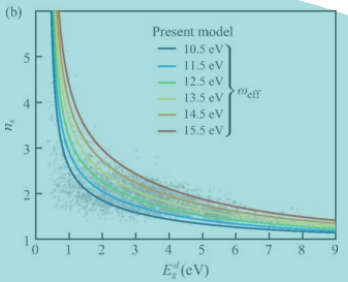
Input generation

Code execution

Error corrections

Parsing outputs

Data storage



Data analysis

Jobflow +

Jobflow-Remote

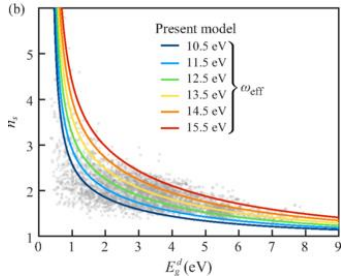
HIGH-THROUGHPUT SIMULATIONS

- High volume of calculated properties/molecules/materials through an automatized process (= Workflow)
- Several steps involved:

Code execution

Error corrections

Data storage



Data analysis

Input generation

Parsing outputs

IO packages

JOBFLOW, JOBFLOW-REMOTE AND ATOMATE2

JOBFLOW

Jobflow is a free, open-source library for writing and executing workflows. Complex workflows can be defined using simple python functions and executed locally or on arbitrary computing resources using the jobflow-remote or FireWorks workflow managers.

Some features that distinguish jobflow are dynamic workflows, easy compositing and connecting of workflows, and the ability to store workflow outputs across multiple databases.

- Clean and flexible Python API
- Integration with multiple databases (MongoDB, AWS S3, GridFS, Azure Blob storage, …)
- Support for dynamic workflows
  - Additions
  - Restarts
- Can support different workflow managers (e.g., Jobflow-Remote, Fireworks)

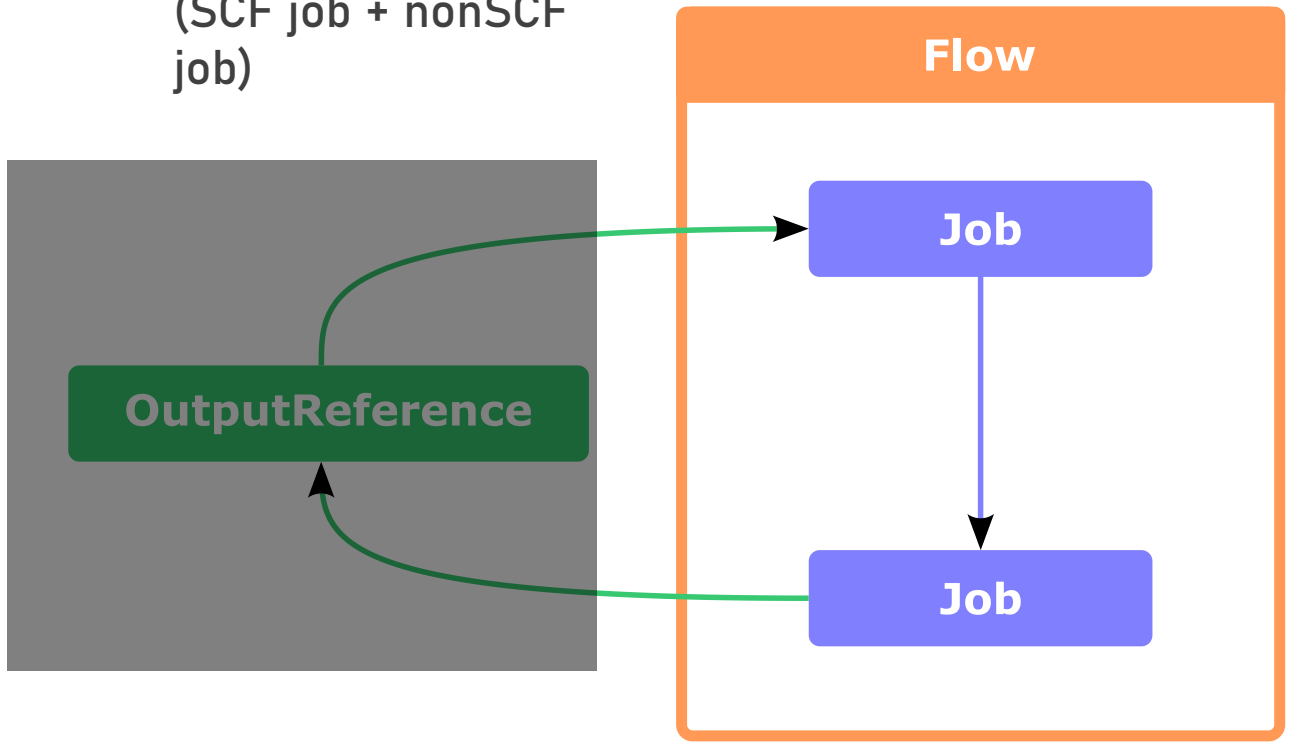
JOBFLOW : CORE CONCEPTS

Job

Is an atomic computing job, e.g. can be an SCF calculation, a Relaxation, …

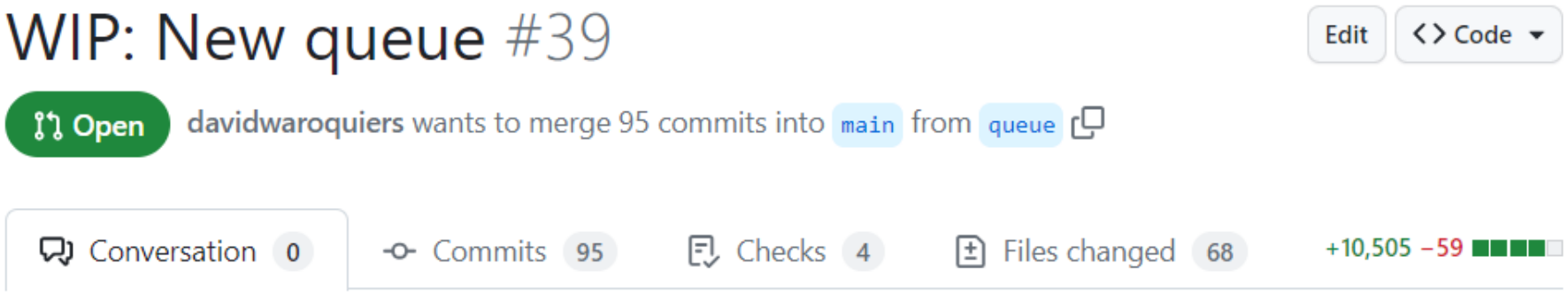
Flow

Is a collection of Jobs or other Flows, e.g. a band structure (SCF job + nonSCF job)

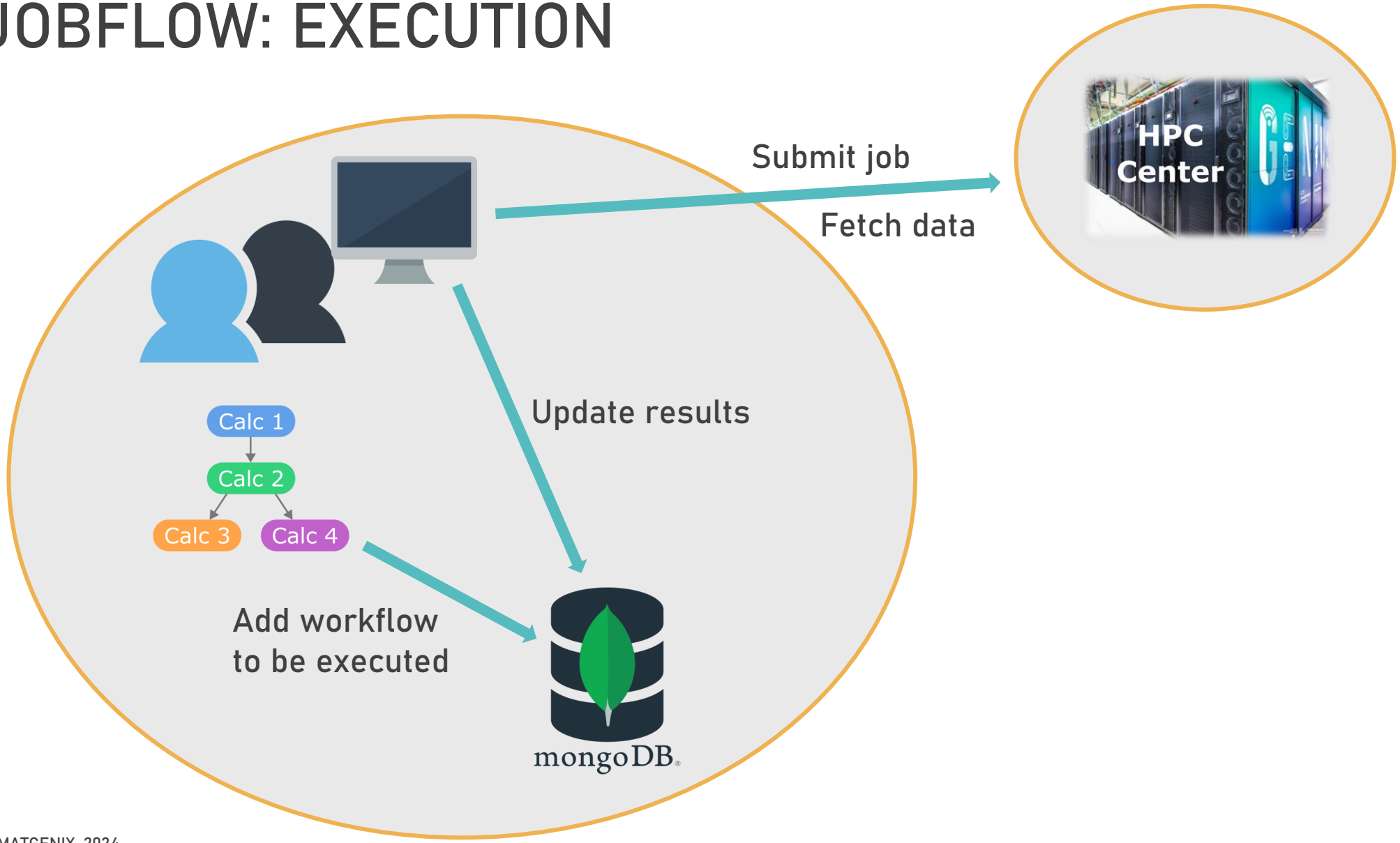


JOBFLOW-REMOTE: EXECUTION

- Developed by Matgenix
- Is the recommended execution engine of jobflow
- Part of project with umicore



JOBFLOW: EXECUTION



ATOMATE2

Atomate2 is a free, open-source software for performing complex materials science workflows using simple Python functions. Features of atomate2 include

- It is built on open-source libraries: pymatgen, custodian, jobflow, and FireWorks.
- A library of "standard" workflows to compute a wide variety of desired materials properties.
- The ability scale from a single material, to 100 materials, or 100,000 materials.
- Easy routes to modifying and chaining workflows together.
- It can build large databases of output properties that you can query, analyze, and share in a systematic way.
- It automatically keeps meticulous records of jobs, their directories, runtime parameters, and more.

- Clean and flexible Python API
- Workflows for VASP, Abinit, CP2K, FHI-AIMS, lobster, qchem, MLIP's, …
  - Add-ons also possible: atomate2-turbomole
- Easily share (parametrized) workflows with a single file representing the workflow
- Can be executed with or without a database (MongoDB)

ATOMATE2

- Based on Jobflow
- Concept of “Makers”

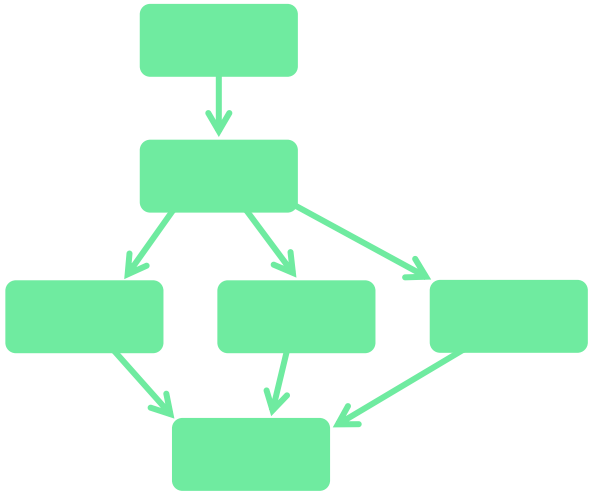
Inputs

Structure, Molecule, charge, …

Maker

Method, functional, basis set, dispersion correction, …

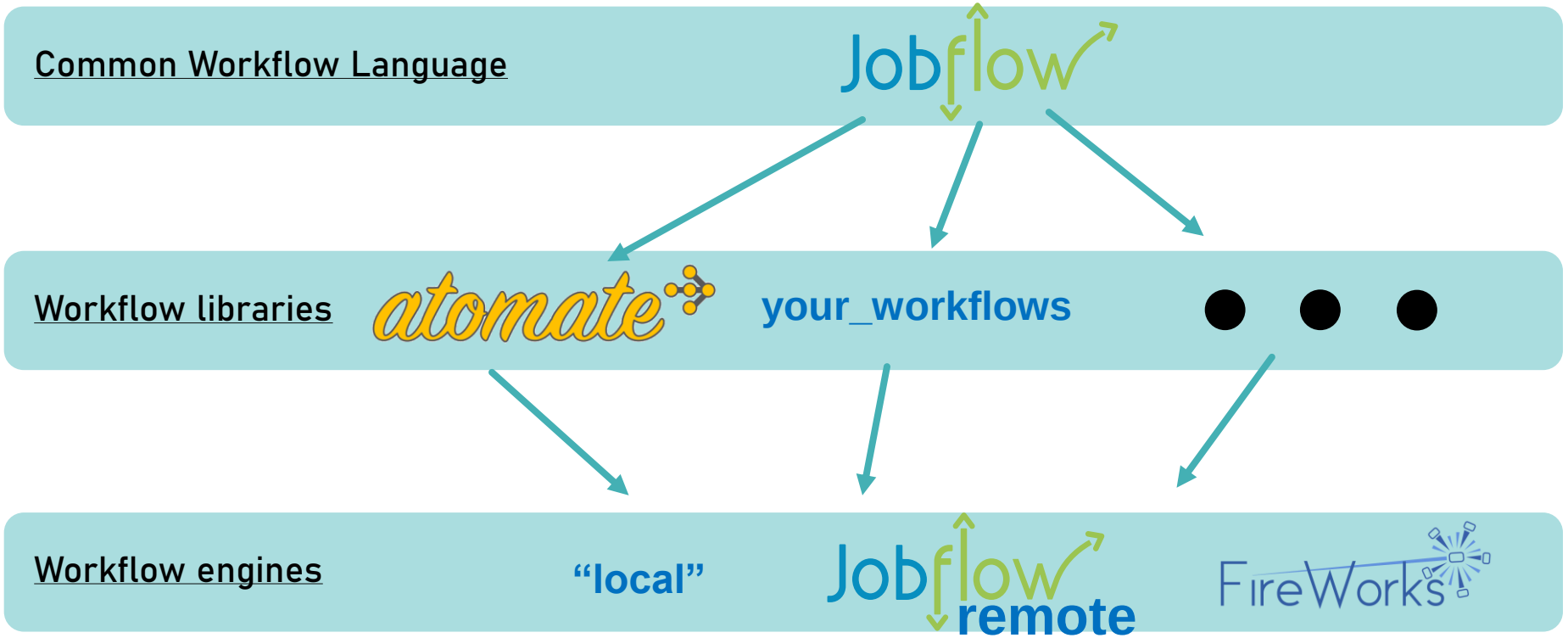
Creates



Workflow

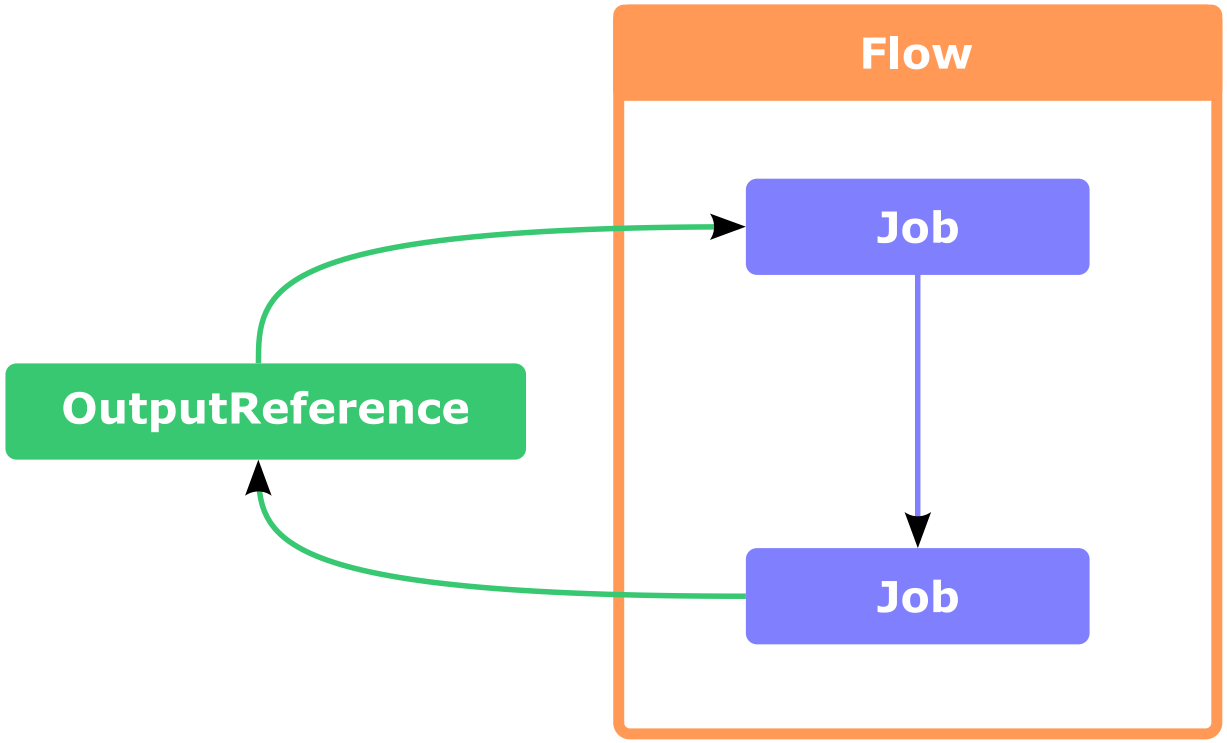
ATOMATE2/JOBFLOW/JOBFLOW_REMOTE

- Jobflow can be seen as a Common Workflow Language
- Workflows developed using jobflow could be executed in any workflow engine (or locally, without any additional package/framework/database)



JOBFLOW

Jobflow CORE CONCEPTS



Jobflow CORE CONCEPTS

Job

- Is an atomic computing job
- Any python function can be a Job
- Must return “serializable” outputs, i.e. outputs that can be easily stored in a database

Flow

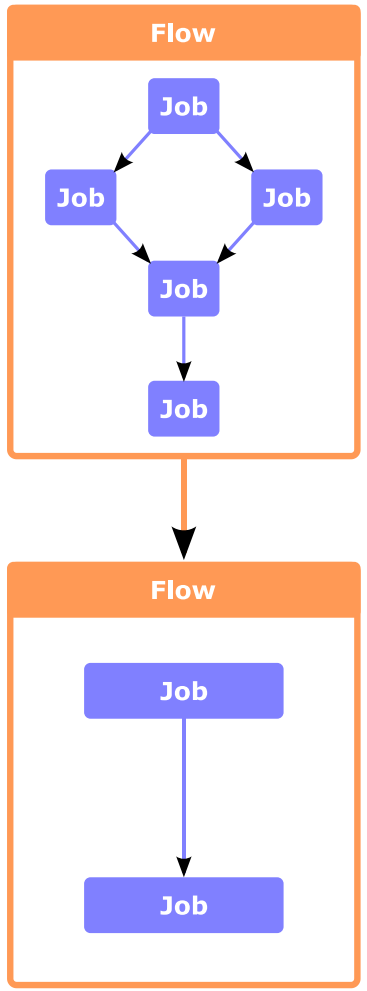
- Is a collection of Jobs or other Flows
- Connectivity between Jobs/Flows is automatically determined from the Jobs and Flows inputs
- Execution order automatically determined from the connectivity

OutputReference

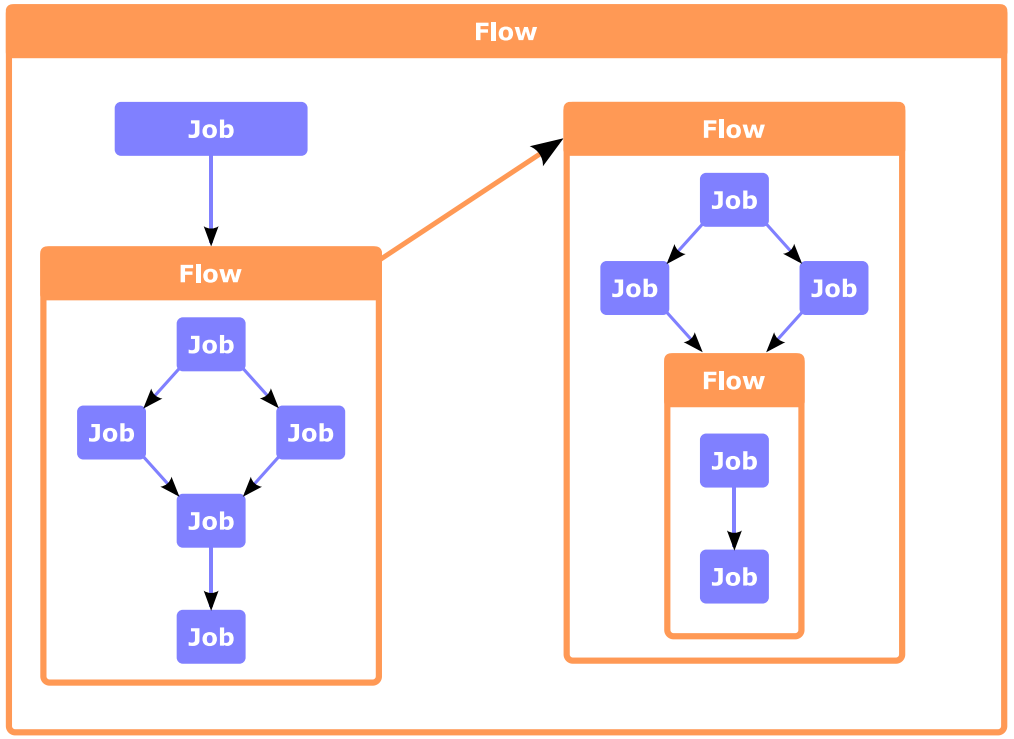
- Deferred output of a Job or a Flow
- Used to construct the workflow graph
- Also allows for more complex actions (e.g. dynamic workflows)

CONNECTING AND COMPOSING

Connecting workflows

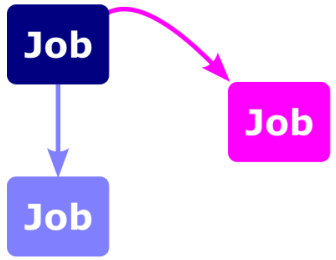


Composed workflows

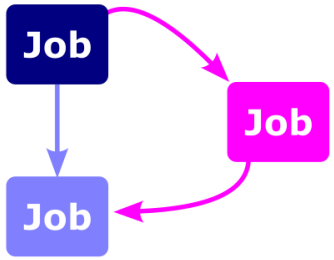


DYNAMIC WORKFLOWS

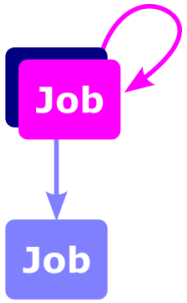
Addition



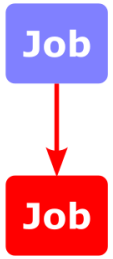
Detour



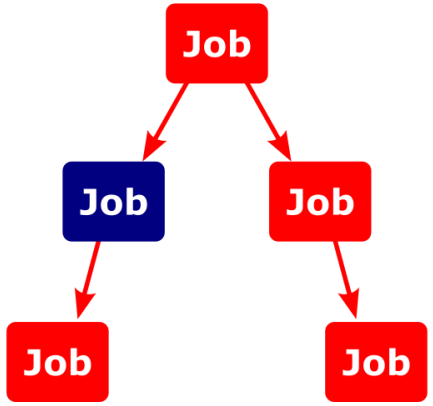
Replace



Stop children



Stop flow



JOBS

- This modifies the actual function/method
  - The method now return a Job object
  - The execution is deferred
  - The function/method should contain what needs to be executed but will be executed later
- Each job has a unique identifier (and also an index for replaced jobs)

```
from jobflow import job

@job
def add(a, b):
    return a + b

add_job = add(1, 2)
```

FLOWS

- A Flow is a list of Jobs with their interdependencies
- Jobs outputs are accessed through the *output* attribute

```python
from jobflow import job, Flow

@job
def add(a, b):
    return a + b

@job
def multiply(a, b):
    return a * b

add_job = add(1, 2)
multiply_job = multiply(add_job.output, 3)

flow = Flow([add_job, multiply_job])
flow.draw_graph().show()
```

FLOWS

- Connectivity/Dependencies automatically detected
- Graph of connectivity easily visualized

```python
from jobflow import job, Flow

@job
def add(a, b):
    return a + b

@job
def multiply(a, b):
    return a * b

add_job = add(1, 2)
multiply_job = multiply(add_job.output, 3)

flow = Flow([add_job, multiply_job])
flow.draw_graph().show()
```

RUNNING JOBS AND FLOWS

- Jobs and Flows can be run locally using the *run_locally* method of jobflow
  - Runs in the current directory and returns the output of the job
  - Reports some logging messages

```
from jobflow import job
from jobflow import run_locally

@job
def add(a, b):
    return a + b

add_job = add(1, 2)
output = run_locally(add_job)
print(output)
```

```
2022-07-26 16:36:17,018 INFO Started executing jobs locally
2022-07-26 16:36:17,135 INFO Starting job - add (340be7e3-21a1-4db4-a0cc-94b3ed039408)
2022-07-26 16:36:17,135 INFO Finished job - add (340be7e3-21a1-4db4-a0cc-94b3ed039408)
2022-07-26 16:36:17,135 INFO Finished executing jobs locally
```

RUNNING JOBS AND FLOWS

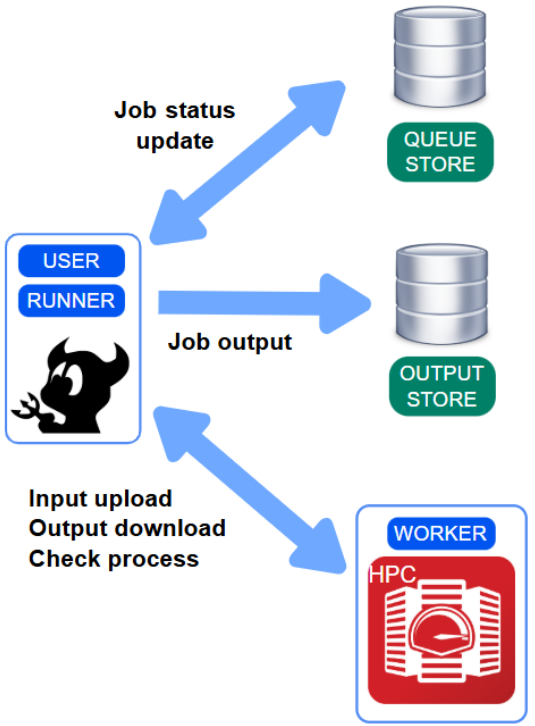
- Jobs and Flows can be run locally using the *run_locally* method of jobflow
  - Runs in the current directory and returns the output of the job
  - Reports some logging messages
  - Ok for testing or few calculations
  - Not ok for production (high-throughput)

JOBFLOW-REMOTE

JOBFLOW-REMOTE

Jobflow-remote is a free, open-source library serving as a manager for the execution of jobflow workflows. While jobflow is not bound to be executed with a specific manager and some adapter has already been developed (*e.g.* Fireworks), jobflow-remote has been designed to take full advantage of and adapt to jobflow's functionalities and interact with the typical high performance computing center accessible by researchers.

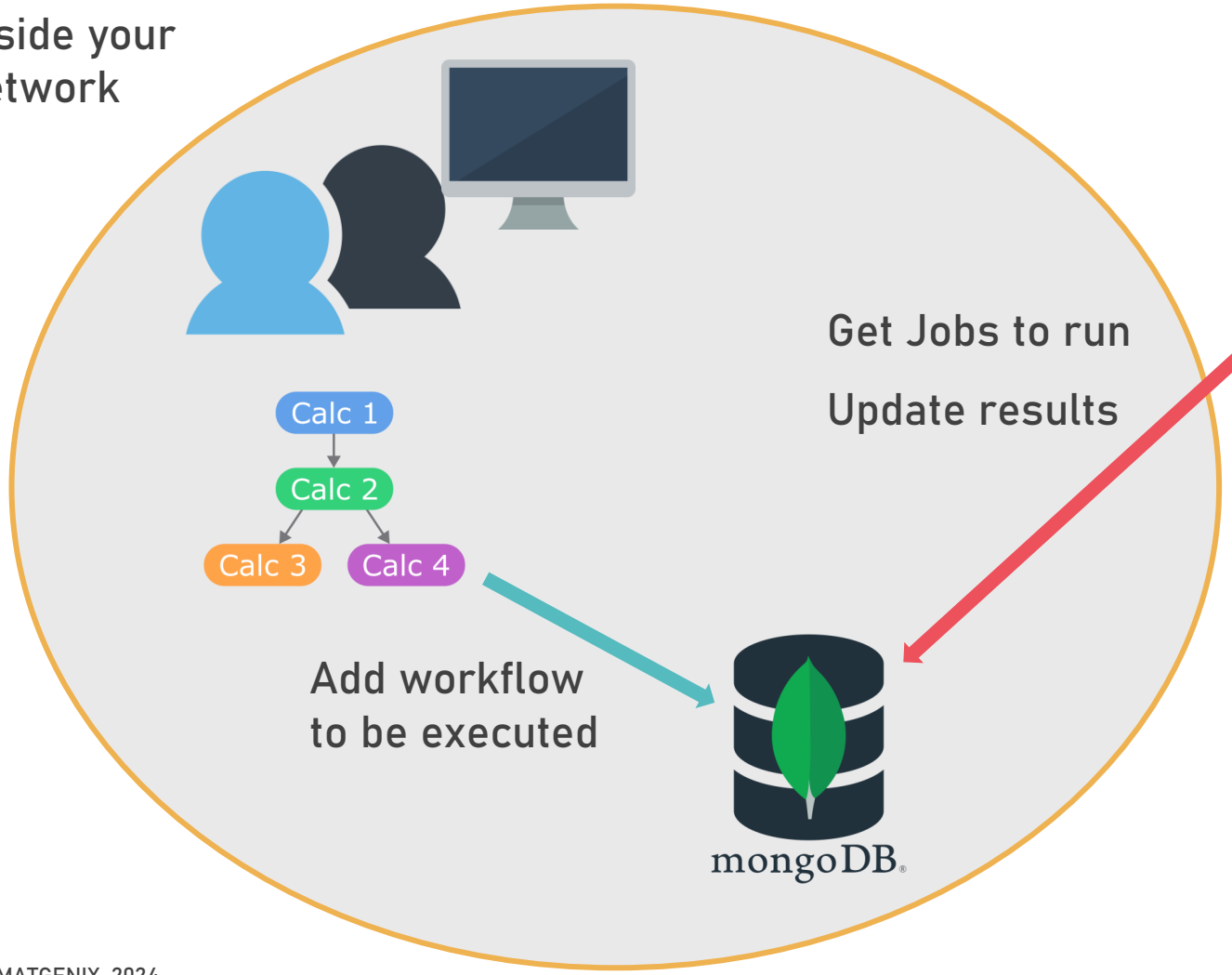
- Fully compatible with jobflow
- Data storage based on mongo-like maggma Stores.
- Simple single file configuration as a starting point. Can scale to handle different projects with different configurations
- Fully configurable submission options
- Management through python API and command line interface
- Parallelized daemon execution
- Limit number of jobs submitted per worker
- Batch submission (experimental)



FIREWORKS EXECUTION

Outside your network

Inside your network



Problem:

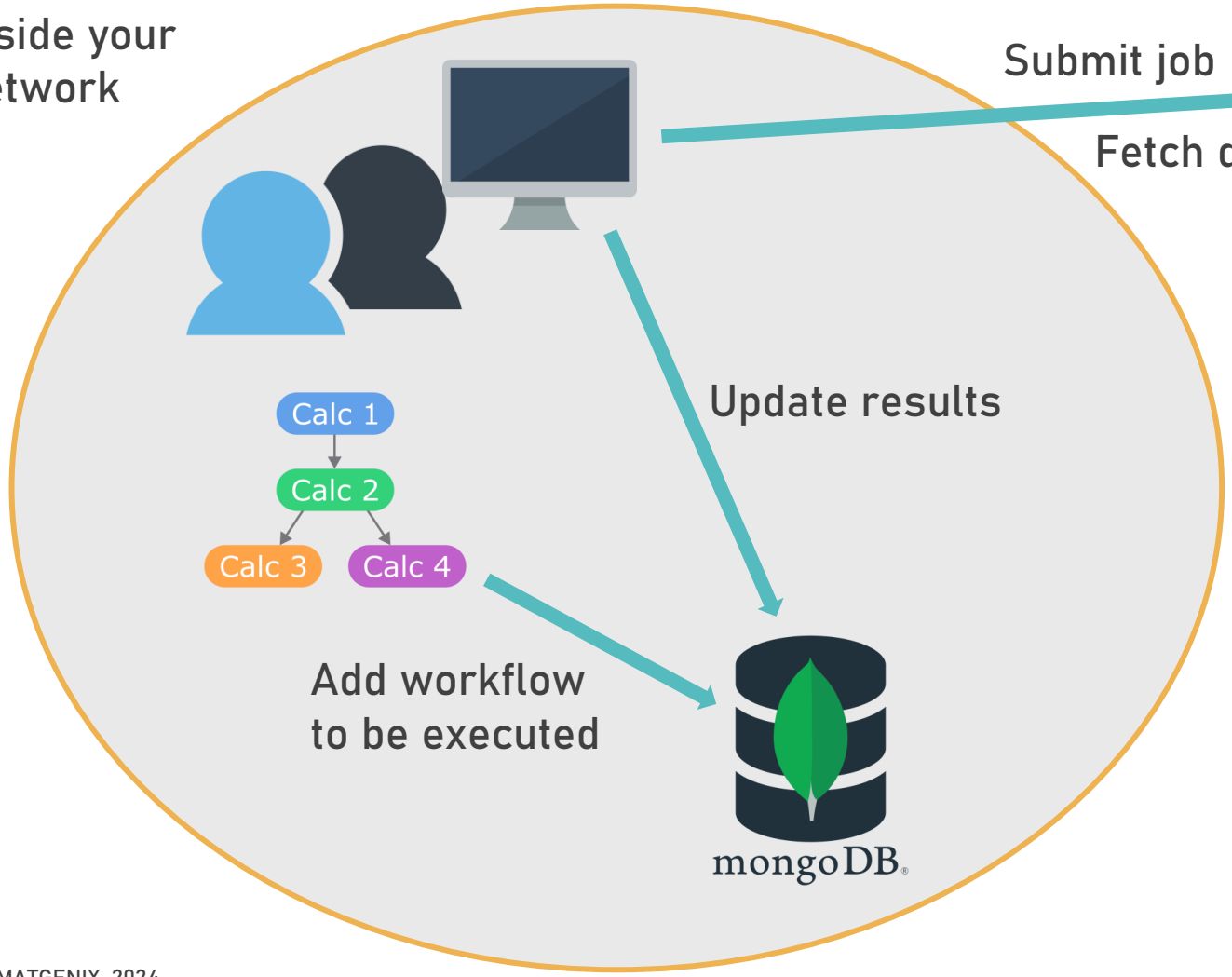
Inbound connections to your own network!

=> Implementation of a remote execution mode

JOBFLOW REMOTE EXECUTION

Inside your network

Outside your network



Solution:

Only outbound connections from your network to the outside

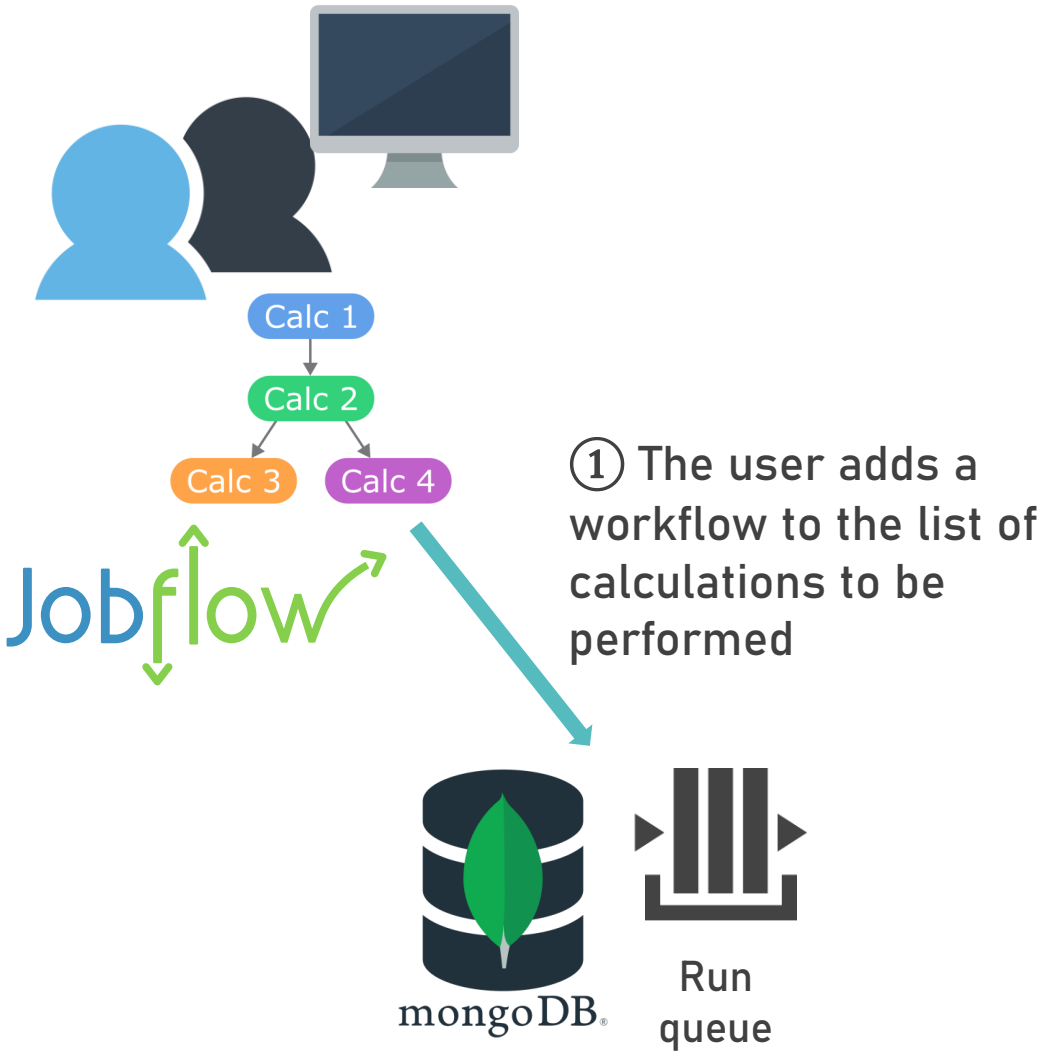
=> Implementation of a jobflow remote mode of execution

WHY JOBFLOW-REMOTE?

Why writing a new manager for jobflow?

- Request from a customer: 
  - Internal DB for results and WF cannot be accessed from the HPC centre
  - Only outbound connections
- Initial implementation extending fireworks
  - hackish
  - Limited functionalities
- Overlapping functionalities in Fireworks and Jobflow
  - workflow definition
- Manager tailored to jobflow

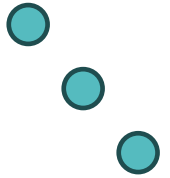
HIGH LEVEL



① The user adds a workflow to the list of calculations to be performed

HIGH LEVEL

② Calculations are submitted to a supercomputer



Run queue

HIGH LEVEL

② Calculations are submitted to a supercomputer or, in the future, to a cloud computing resource

Run queue

HIGH LEVEL

③ Results are brought back by the runner, …

HIGH LEVEL

③ Results are brought back by the runner, …

and inserted into the database

Maggma

Job Store

Database containing the standardized outputs of the calculations

HIGH LEVEL

③ Results are brought back by the runner, …

and inserted into the database

Maggma

Large data/files are/can be stored in different type of storage

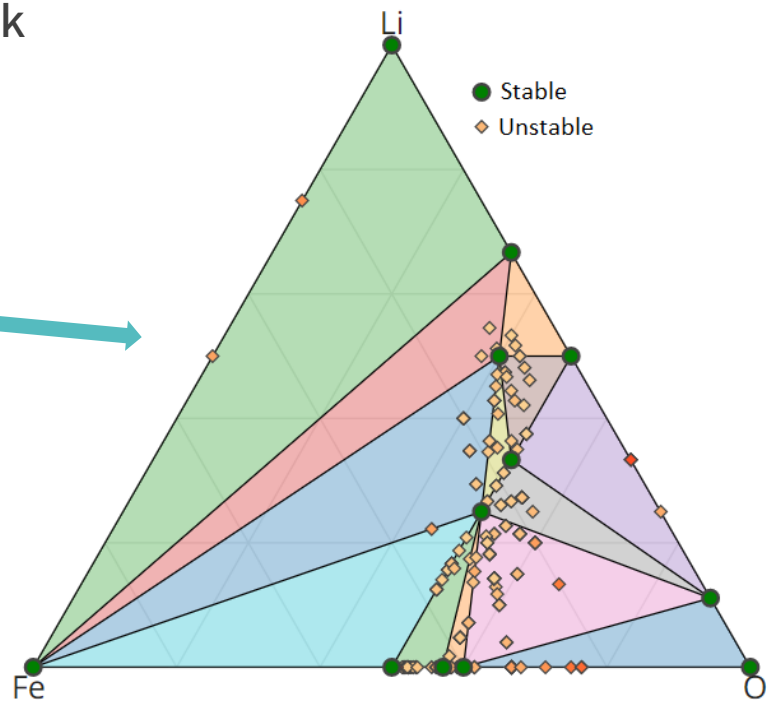
Azure Blobs

Amazon S3

mongoDB.
GridFS

HIGH LEVEL

④ The user can access the results from the work station/virtual machine and perform analysis, visualizations, …



Maggma

mongoDB.

Search...

Job Store

Azure Blobs

Amazon S3

mongoDB.
GridFS

JOBFLOW REMOTE: COMMAND LINE INTERFACE

```
(atomate2_mlff) david@mtgx-23-007:~$ jf

 Usage: jf [OPTIONS] COMMAND [ARGS]...

 The controller CLI for jobflow-remote

╭─ Options ──────────────────────────────────────────────────────────────────╮
│ --project    -p       TEXT  Select a project for the current             │
│                             execution                                     │
│                             [default: None]                               │
│ --full-exc   -fe            Print the full stack trace of exception       │
│                             when enabled                                  │
│ --help       -h             Show this message and exit.                   │
╰────────────────────────────────────────────────────────────────────────────╯
╭─ Commands ─────────────────────────────────────────────────────────────────╮
│ admin     Commands for administering the database                          │
│ flow      Commands for managing the flows                                  │
│ job       Commands for managing the jobs                                   │
│ project   Commands concerning the project definition                       │
│ runner    Commands for handling the Runner                                 │
╰────────────────────────────────────────────────────────────────────────────╯
```

- jf command
  - Help for all the commands
- What can you do ?
  - Get information about the project(s)
  - Get information about the jobs or flows
  - Create/manage projects
  - Deal with the runner daemon

DATABASE OF CALCULATIONS

```
(turbomole_workflows) david@mtgx-23-007:~$ jf flow list -m 20
The selected project is demo_tm_workflows from config file
/home/david/.jfremote/demo_tm_workflows.yaml
The number of Flows printed is limited by the maximum selected: 20
```

Flows info

DB id	Name	State	Flow id	Num Jobs	Last updated [CET]
673	Flow	COMPLETED	a97b49d5-8111-4495-9…	3	2024-09-16 23:34
670	Flow	COMPLETED	a47d951c-8f9e-41c0-b…	3	2024-09-16 23:34
667	Flow	COMPLETED	795c1c5c-0f50-4be6-b…	3	2024-09-16 23:34
661	Flow	COMPLETED	d4d2b7c1-7b95-4099-b…	3	2024-09-16 23:34
658	Flow	COMPLETED	8208e6b6-c703-41c2-8…	3	2024-09-16 23:34
733	Flow	COMPLETED	b32e7aab-b281-4522-8…	3	2024-09-16 23:33
718	Flow	COMPLETED	529976b0-c97d-4c4c-a…	3	2024-09-16 23:33
709	Flow	COMPLETED	db1da777-35f0-4be3-9…	3	2024-09-16 23:33
682	Flow	COMPLETED	8a999e19-e068-48f1-a…	3	2024-09-16 23:33
655	Flow	COMPLETED	ea0a9f37-45ea-4594-a…	3	2024-09-16 23:32
652	Flow	COMPLETED	a0fcec61-1e32-4a49-9…	3	2024-09-16 23:32
649	Flow	COMPLETED	d81a7491-2bd2-41d2-8…	3	2024-09-16 23:32
646	Flow	COMPLETED	2d22c46c-6198-4128-b…	3	2024-09-16 23:32
643	Flow	COMPLETED	f6c3bbb3-49af-4d63-9…	3	2024-09-16 23:32
619	Flow	COMPLETED	a0cedd6d-4a22-4ea9-8…	3	2024-09-16 23:27
622	Flow	COMPLETED	8a72175c-2b3d-4ac1-9…	3	2024-09-16 23:27
613	Flow	COMPLETED	6401ee97-5910-4771-8…	3	2024-09-16 23:27
589	Flow	COMPLETED	51e3a441-23d5-4a9c-a…	3	2024-09-16 23:27
583	Flow	COMPLETED	91408c0c-9862-4a98-8…	3	2024-09-16 23:27
637	Flow	COMPLETED	43fc2032-9bd1-489a-a…	3	2024-09-16 23:26

SOON

- Graphical user interface
  - Simple web application
  - Query and look at Flows and Jobs
  - Report: statistics on Flows and Jobs
  - Restart capabilities
- Many other features
- Anything you'd find interesting ?
  - You can open an issue on github

DOCUMENTATION

- Jobflow
  - Documentation: https://materialsproject.github.io/jobflow/
  - Github: https://github.com/materialsproject/jobflow
- Jobflow-Remote
  - Documentation: https://matgenix.github.io/jobflow-remote/index.html
  - Github: https://github.com/Matgenix/jobflow-remote
- Atomate2
  - Documentation: https://materialsproject.github.io/atomate2/
  - Github: https://github.com/materialsproject/atomate2

CONCLUSIONS

- Why do we need workflows ?
- Workflows made simple with 

- Executing workflows remotely on multiple supercomputers with 

Matthew Evans, Phd

Prof. Geoffroy Hautier

Prof. Gian-Marco Rignanese

Matgenix

Guido Petretto, PhD

Guillaume Brunin, PhD

David Waroquiers, PhD

And our newly joined recruits: Gabriel Taillandier, Francesco Ricci, Julien Bouquiaux

Matgenix

David Waroquiers, CEO and Co-Founder

david.waroquiers@matgenix.com

https://matgenix.com/

https://www.linkedin.com/in/david-waroquiers/

RUNNING JOBS AND FLOWS

- Jobs can be run locally using the *run_locally* method of jobflow
  - Runs in the current directory and returns the output of the job
  - Reports some logging messages

```
from jobflow import job
from jobflow import run_locally

@job
def add(a, b):
    return a + b

add_job = add(1, 2)
output = run_locally(add_job)
print(output)
```

```
2022-07-26 16:36:17,018 INFO Started executing jobs locally
2022-07-26 16:36:17,135 INFO Starting job - add (340be7e3-21a1-4db4-a0cc-94b3ed039408)
2022-07-26 16:36:17,135 INFO Finished job - add (340be7e3-21a1-4db4-a0cc-94b3ed039408)
2022-07-26 16:36:17,135 INFO Finished executing jobs locally
```

RUNNING JOBS AND FLOWS

- Jobs can be run locally using the *run_locally* method of jobflow
  - Runs in the current directory and returns the output of the job
  - Reports some logging messages

```
from jobflow import job

@job
def add(a, b):
    return a + b

add_job = add(1, 2)
output = run_locally(add_job)
print(output)
```

```
{'c91d0183-3121-4b4f-ab11-ea67a26d534f': {1: Response(output=3,
                                          detour=None,
                                          addition=None,
                                          replace=None,
                                          stored_data=None,
                                          stop_children=False,
                                          stop_jobflow=False)}}
```

RUNNING JOBS AND FLOWS

- Same with Flows
  - Returns the output of all the jobs

```
from jobflow import job, Flow, run_locally

@job
def add(a, b):
    return a + b

@job
def multiply(a, b):
    return a * b

add_job = add(1, 2)
multiply_job = multiply(add_job.output, 3)

flow = Flow([add_job, multiply_job])
output = run_locally(flow)
print(output)
```

RUNNING JOBS AND FLOWS

- Same with Flows
  - Logging

```
2022-07-26 16:44:32,873 INFO Started executing jobs locally
2022-07-26 16:44:32,874 INFO Starting job - add (4f75f404-4ee0-4dc0-95cd-f9a1d18c123f)
2022-07-26 16:44:32,874 INFO Finished job - add (4f75f404-4ee0-4dc0-95cd-f9a1d18c123f)
2022-07-26 16:44:32,874 INFO Starting job - multiply (4b47282e-4a71-43b6-9e63-fe9d876c015a)
2022-07-26 16:44:32,875 INFO Finished job - multiply (4b47282e-4a71-43b6-9e63-fe9d876c015a)
2022-07-26 16:44:32,875 INFO Finished executing jobs locally
```

RUNNING JOBS AND FLOWS

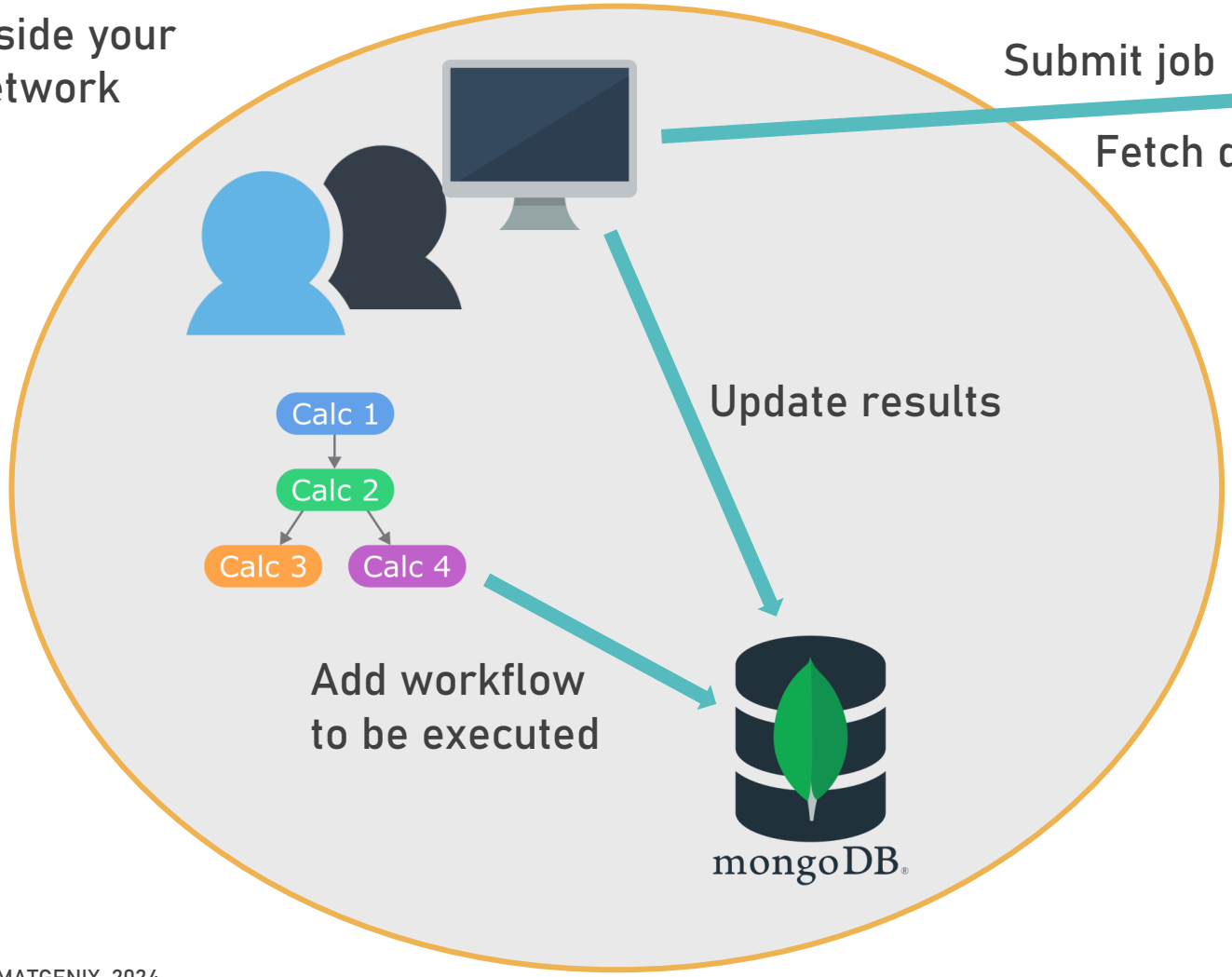
- Same with Flows
  - Logging
  - Output of all the jobs

```
{'4b47282e-4a71-43b6-9e63-fe9d876c015a': {1: Response(output=9,
                                                      detour=None,
                                                      addition=None,
                                                      replace=None,
                                                      stored_data=None,
                                                      stop_children=False,
                                                      stop_jobflow=False)},
 '4f75f404-4ee0-4dc0-95cd-f9a1d18c123f': {1: Response(output=3,
                                                      detour=None,
                                                      addition=None,
                                                      replace=None,
                                                      stored_data=None,
                                                      stop_children=False,
                                                      stop_jobflow=False)}}
```

JOBFLOW REMOTE EXECUTION

Inside your network

Outside your network



Solution:

Only outbound connections from your network to the outside

=> Implementation of a jobflow remote mode of execution

TESTIMONIALS

Just a quick note to say that I've started using jobflow remote in a medium throughput project (~200 calculations) and I'm so impressed.

You've done an incredible job with it. It feels very natural to use and there are so many useful options.

Alex Ganose, main developer of jobflow

I've never run high-throughput computations more conveniently.

Janine George

Thanks, you guys are awesome. I was telling Joseph and Jorge last week I'm very glad we went with jf-remote over fireworks.

It's leaner, better maintained and feels like a version 2 tool similar to how pytorch feels much more polished than tensorflow.

Janosh Riebesell, Radical AI

WHY JOBFLOW-REMOTE?

Why writing a new manager for jobflow?

- Request from a customer: 
  - Internal DB for results and WF cannot be accessed from the HPC centre
  - Only outbound connections
- Initial implementation extending fireworks
  - hackish
  - Limited functionalities
- Overlapping functionalities in Fireworks and Jobflow
  - workflow definition
- Manager tailored to jobflow

MAIN FEATURES

- Manage the state of Jobs and Flows
- Job execution does not need access to the DB
- Daemon process orchestrating Jobs execution
  - Handles multiple “workers” (supercomputer, local execution, supercomputer frontend, …)
- Retries, restarts (with fail-safe mechanisms)
- Extensive CLI, programmatic API
- Optional multiple projects
- Different setups allowed

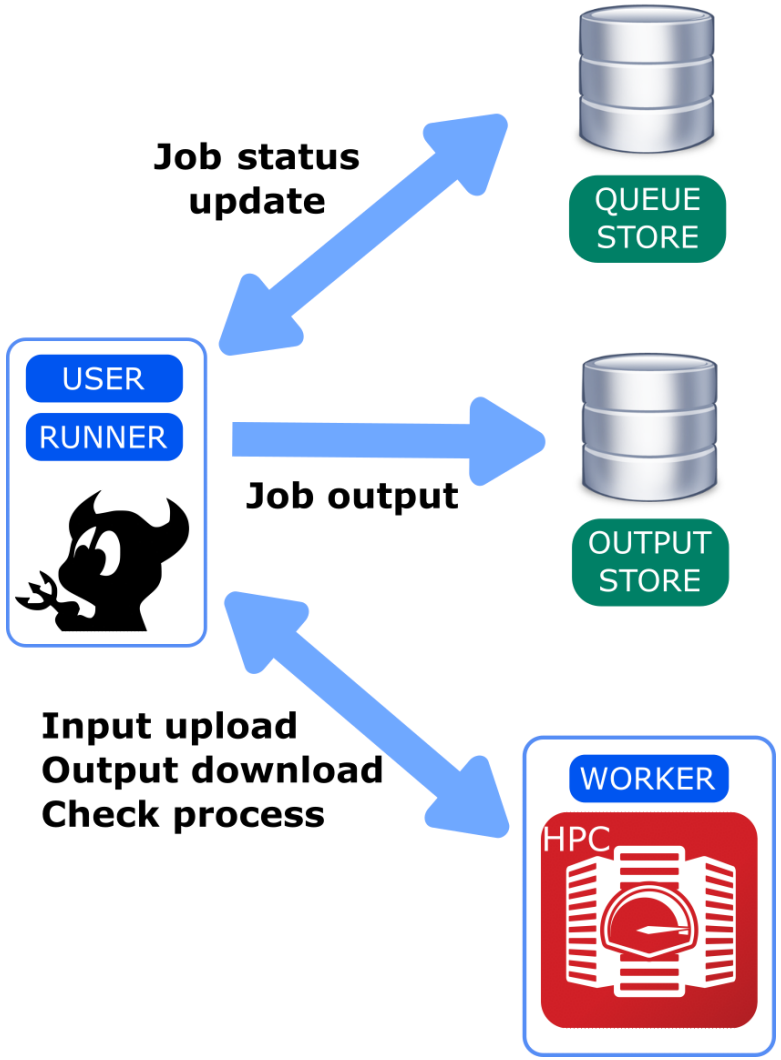
Experimental

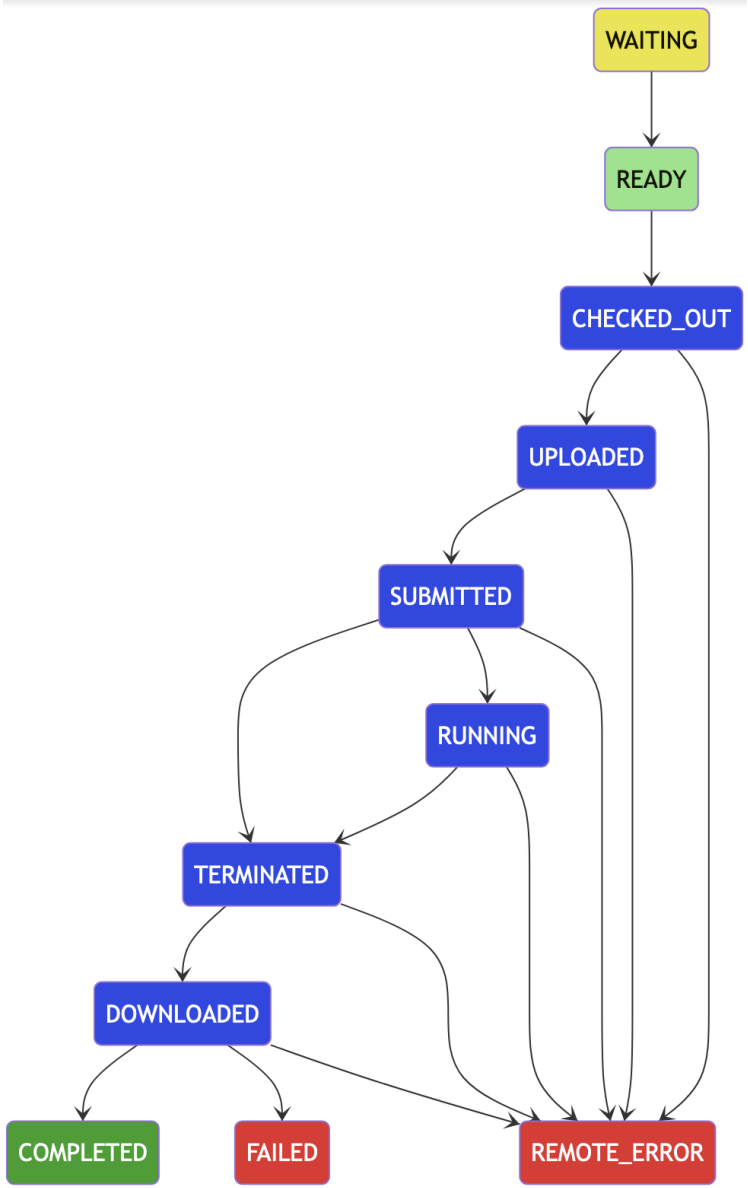
- Batch submission
  - Similar to Firework’s rapidfire with 1 Job
  - N parallel Jobs will be implemented
- Connection with OTP

Development

- Open-source
- Integration tests (ongoing): real mongodb database using docker containers

WORKING PRINCIPLES

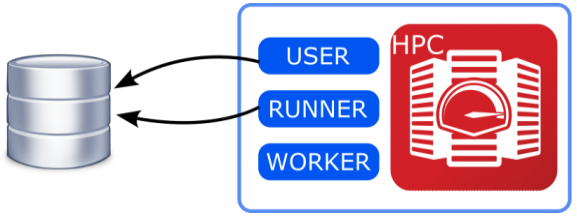




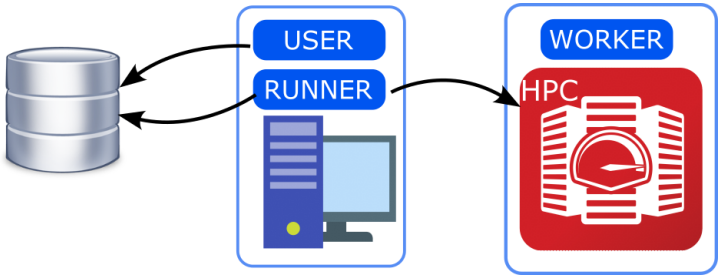
CONFIGURATION

POSSIBLE CONFIGURATIONS

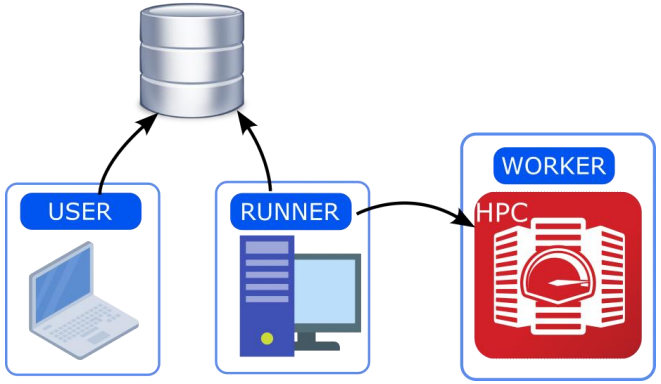
- All-in-one
  - Running completely on the cluster



- User-Workstation
  - A workstation hosting the daemon and used for user interactions



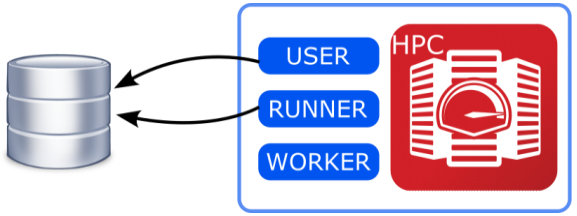
- Full split
  - Workstation for the daemon and separate system for user interaction



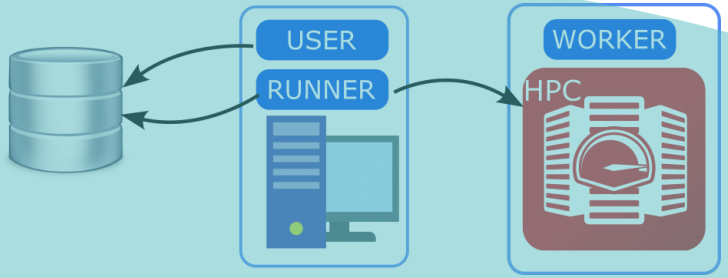
The same python environment should be present on all the machines

POSSIBLE CONFIGURATIONS

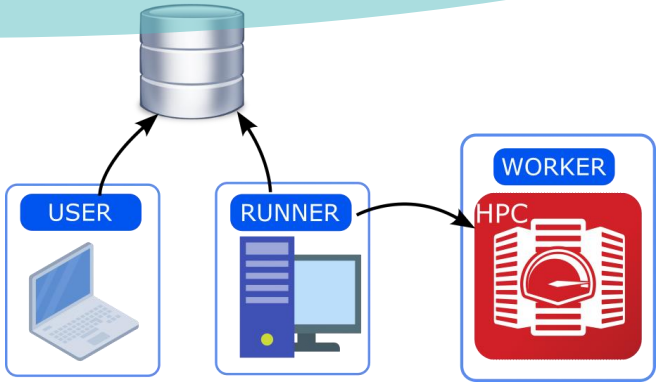
- All-in-one
  - Running completely on the cluster



- User-Workstation
  - A workstation hosting the daemon and used for user interactions



- Full split
  - Workstation for the daemon and separate system for user interaction



The same python environment should be present on all the machines

CREATE A PROJECT

```
(atomate2_mlff) david@mtgx-23-007:~$ jf project generate -h

 Usage: jf project generate [OPTIONS] NAME

 Generate a project configuration file with dummy elements to be
 edited manually

╭─ Arguments ──────────────────────────────────────────────────────╮
*    name       TEXT  Name of the project [default: None]
                      [required]
╰──────────────────────────────────────────────────────────────────╯
╭─ Options ────────────────────────────────────────────────────────╮
--format  -f      [json|yaml|toml]  File format [default: yaml]
--full                              Generate a configuration file
                                    with all the fields and more
                                    elements
--help    -h                        Show this message and exit.
╰──────────────────────────────────────────────────────────────────╯
```

CREATE A PROJECT

```
(atomate2_mlff) david@mtgx-23-007:~$ jf project generate czts
Configuration file for project czts created in
/home/david/.jfremote/czts.yaml
```

Content of the czts.yaml file

```yaml
name: czts
workers:
  example_worker:
    type: remote
    scheduler_type: slurm
    work_dir: /path/to/run/folder
    pre_run: source /path/to/python/environment/activate
    timeout_execute: 60
    host: remote.host.net
    user: bob
queue:
  store:
    type: MongoStore
    host: localhost
    database: db_name
    username: bob
    password: secret_password
    collection_name: jobs
exec_config: {}
```

```yaml
jobstore:
  docs_store:
    type: MongoStore
    database: db_name
    host: host.mongodb.com
    port: 27017
    username: bob
    password: secret_password
    collection_name: outputs
  additional_stores:
    data:
      type: GridFSStore
      database: db_name
      host: host.mongodb.com
      port: 27017
      username: bob
      password: secret_password
      collection_name: outputs_blobs
```

CREATE A PROJECT

```
(atomate2_mlff) david@mtgx-23-007:~$ jf project generate czts
Configuration file for project czts created in
/home/david/.jfremote/czts.yaml
```

Content of the czts.yaml file

STORES IN JOBFLOW

The Job store is configured in the *jobflow.yaml* file.

The store can be accessed in python through the jobflow's settings:

Maggma

```python
from jobflow import SETTINGS

store = SETTINGS.JOB_STORE
```

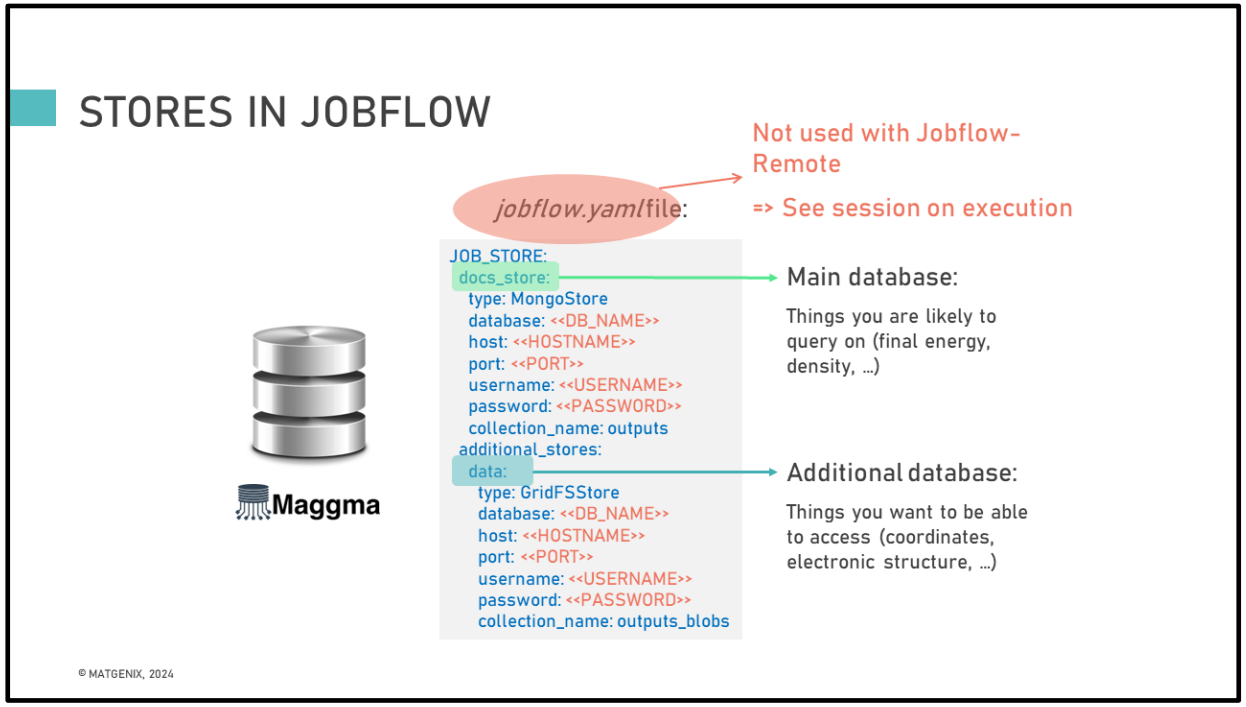


```yaml
jobstore:
  docs_store:
    type: MongoStore
    database: db_name
    host: host.mongodb.com
    port: 27017
    username: bob
    password: secret_password
    collection_name: outputs
  additional_stores:
    data:
      type: GridFSStore
      database: db_name
      host: host.mongodb.com
      port: 27017
      username: bob
      password: secret_password
      collection_name: outputs_blobs
```

CREATE A PROJECT

```
(atomate2_mlff) david@mtgx-23-007:~$ jf project generate czts
Configuration file for project czts created in
/home/david/.jfremote/czts.yaml
```

Content of the czts.yaml file




```yaml
jobstore:
  docs_store:
    type: MongoStore
    database: db_name
    host: host.mongodb.com
    port: 27017
    username: bob
    password: secret_password
    collection_name: outputs
  additional_stores:
    data:
      type: GridFSStore
      database: db_name
      host: host.mongodb.com
      port: 27017
      username: bob
      password: secret_password
      collection_name: outputs_blobs
```

CREATE A PROJECT

```
(atomate2_mlff) david@mtgx-23-007:~$ jf project generate czts
Configuration file for project czts created in
/home/david/.jfremote/czts.yaml
```

Content of the czts.yaml file

```yaml
name: czts
workers:
  example_worker:
    type: remote
    scheduler_type: slurm
    work_dir: /path/to/run/folder
    pre_run: source /path/to/python/environment/activate
    timeout_execute: 60
    host: remote.host.net
    user: bob
queue:
  store:
    type: MongoStore
    host: localhost
    database: db_name
    username: bob
    password: secret_password
    collection_name: jobs
exec_config: {}
```

Supercomputer to run the workflows

Database with the dependencies and Job/Flow states

CREATE A PROJECT

```
(atomate2_mlff) david@mtgx-23-007:~$ jf project generate czts --full
Configuration file for project czts created in
/home/david/.jfremote/czts.yaml
```

```yaml
name: czts
base_dir: /home/david/.jfremote/czts
tmp_dir: /home/david/.jfremote/czts/tmp
log_dir: /home/david/.jfremote/czts/log
daemon_dir: /home/david/.jfremote/czts/daemon
log_level: info
runner:
  delay_checkout: 30
  delay_check_run_status: 30
  delay_advance_status: 30
  delay_refresh_limited: 600
  delay_update_batch: 60
  lock_timeout: 86400
  delete_tmp_folder: true
  max_step_attempts: 3
  delta_retry:
  - 30
  - 300
  - 1200
```

```yaml
workers:
  example_worker:
    type: remote
    scheduler_type: slurm
    work_dir: /path/to/run/folder
    resources:
    pre_run: source /path/to/python/environment/activate
    post_run:
    timeout_execute: 60
    max_jobs:
    batch:
    host: remote.host.net
    user: bob
    port:
    password:
    key_filename:
    passphrase:
    gateway:
    forward_agent:
    connect_timeout:
    connect_kwargs:
    inline_ssh_env:
    keepalive: 60
    shell_cmd: bash
    login_shell: true
    interactive_login: false
```

```yaml
  example_local:
    type: local
    scheduler_type: shell
    work_dir: /path/to/run/folder
    resources:
    pre_run: source /path/to/python/environment/activate
    post_run:
    timeout_execute: 60
    max_jobs:
    batch:
queue:
  store:
    type: MongoStore
    host: localhost
    database: db_name
    username: bob
    password: secret_password
    collection_name: jobs
  flows_collection: flows
  auxiliary_collection: jf_auxiliary
  db_id_prefix:
```

CREATE A PROJECT

```
(atomate2_mlff) david@mtgx-23-007:~$ jf project generate czts --full
Configuration file for project czts created in
/home/david/.jfremote/czts.yaml
```

```yaml
exec_config:
  example_config:
    modules:
    - GCC/10.2.0
    - OpenMPI/4.0.5-GCC-10.2.0
    export:
      PATH: /path/to/binaries:$PATH
    pre_run: conda activate env_name
    post_run:
```

```yaml
jobstore:
  docs_store:
    type: MongoStore
    database: db_name
    host: host.mongodb.com
    port: 27017
    username: bob
    password: secret_password
    collection_name: outputs
  additional_stores:
    data:
      type: GridFSStore
      database: db_name
      host: host.mongodb.com
      port: 27017
      username: bob
      password: secret_password
      collection_name: outputs_blobs
remote_jobstore:
metadata:
```

PROJECT CONFIGURATION OPTIONS

Project	
name	str
base_dir	Optional[str]
tmp_dir	Optional[str]
log_dir	Optional[str]
daemon_dir	Optional[str]
log_level	LogLevel(str, Enum)
runner	RunnerOptions
workers	dict[str, Annotated[Union[LocalWorker, RemoteWorker], FieldInfo(annotation=NoneType, required=True, discriminator='type')]]
queue	QueueConfig
exec_config	dict[str, ExecutionConfig]
jobstore	dict
metadata	Optional[dict]

RunnerOptions	
delay_checkout	int
delay_check_run_status	int
delay_advance_status	int
delay_refresh_limited	int
delay_update_batch	int
lock_timeout	Optional[int]
delete_tmp_folder	bool
max_step_attempts	int
delta_retry	tuple[int, ...]

LocalWorker	
type	Literal['local']
scheduler_type	str
work_dir	Path
resources	Optional[dict]
pre_run	Optional[str]
post_run	Optional[str]
timeout_execute	int
max_jobs	Optional[int]
batch	Optional[BatchConfig]

RemoteWorker	
type	Literal['remote']
scheduler_type	str
work_dir	Path
resources	Optional[dict]
pre_run	Optional[str]
post_run	Optional[str]
timeout_execute	int
max_jobs	Optional[int]
batch	Optional[BatchConfig]
host	str
user	Optional[str]
port	Optional[int]
password	Optional[str]
key_filename	Optional[str, list[str]]
passphrase	Optional[str]
gateway	Optional[str]
forward_agent	Optional[bool]
connect_timeout	Optional[int]
connect_kwargs	Optional[dict]
inline_ssh_env	Optional[bool]
keepalive	Optional[int]
shell_cmd	Optional[str]
login_shell	bool

BatchConfig	
jobs_handle_dir	Path
work_dir	Path
max_jobs	Optional[int]
max_wait	Optional[int]
max_time	Optional[int]

QueueConfig	
store	dict
flows_collection	str
auxiliary_collection	str
db_id_prefix	Optional[str]

ExecutionConfig	
modules	Optional[list[str]]
export	Optional[dict[str, Any]]
pre_run	Optional[str]
post_run	Optional[str]

Created by erdantic v0.6.0 <https://github.com/drivendataorg/erdantic>

Project configuration:
- Databases (queue and outputs)
- Execution configurations
- Runner options
- Workers

https://matgenix.github.io/jobflow-remote/user/projectconf.html#project-options

JOBFLOW REMOTE COMMAND LINE

JOBFLOW REMOTE: COMMAND LINE INTERFACE

```
(atomate2_mlff) david@mtgx-23-007:~$ jf

 Usage: jf [OPTIONS] COMMAND [ARGS]...

 The controller CLI for jobflow-remote

╭─ Options ──────────────────────────────────────────────────────────────────╮
│ --project    -p       TEXT  Select a project for the current             │
│                             execution                                     │
│                             [default: None]                               │
│ --full-exc   -fe            Print the full stack trace of exception       │
│                             when enabled                                  │
│ --help       -h             Show this message and exit.                   │
╰────────────────────────────────────────────────────────────────────────────╯
╭─ Commands ─────────────────────────────────────────────────────────────────╮
│ admin     Commands for administering the database                          │
│ flow      Commands for managing the flows                                  │
│ job       Commands for managing the jobs                                   │
│ project   Commands concerning the project definition                       │
│ runner    Commands for handling the Runner                                 │
╰────────────────────────────────────────────────────────────────────────────╯
```

- jf command
  - Help for all the commands
- What can you do ?
  - Get information about the project(s)
  - Get information about the jobs or flows
  - Create/manage projects
  - Deal with the runner daemon

JOBFLOW REMOTE: COMMAND LINE INTERFACE

```
(atomate2_mlff) david@mtgx-23-007:~$ jf runner status
The active project could not be determined and it is required to
execute this command
```

- Project selection
  - -p option in cmd line
  - Environment variable:
    - export JFREMOTE_PROJECT=czts

```
(atomate2_mlff) david@mtgx-23-007:~$ jf -p czts runner status
The selected project is czts from config file
/home/david/.jfremote/czts.yaml
Daemon status: shut_down
```

- jf command
  - Help for all the commands
- What can you do ?
  - Get information about the project(s)
  - Get information about the jobs or flows
  - Create/manage projects
  - Deal with the runner daemon

JOBFLOW REMOTE: COMMAND LINE INTERFACE

```
(atomate2_mlff) david@mtgx-23-007:~$ export JFREMOTE_PROJECT=czts
(atomate2_mlff) david@mtgx-23-007:~$ jf runner status
The selected project is czts from config file
/home/david/.jfremote/czts.yaml
Daemon status: shut_down
```

```
(atomate2_mlff) david@mtgx-23-007:~$ jf runner start
The selected project is czts from config file
/home/david/.jfremote/czts.yaml
(atomate2_mlff) david@mtgx-23-007:~$ jf runner status
The selected project is czts from config file
/home/david/.jfremote/czts.yaml
Daemon status: running
```

- jf command
  - Help for all the commands
- What can you do ?
  - Get information about the project(s)
  - Get information about the jobs or flows
  - Create/manage projects
  - Deal with the runner daemon

JOBFLOW REMOTE: COMMAND LINE INTERFACE

```
(atomate2_mlff) david@mtgx-23-007:~$ jf project list
The selected project is czts from config file
/home/david/.jfremote/czts.yaml
List of projects in /home/david/.jfremote
 - catalysis
 - czts
 - demo_project
 - tata
```

```
(atomate2_mlff) david@mtgx-23-007:~$ jf project check
The selected project is czts from config file
/home/david/.jfremote/czts.yaml
✓ Worker local
✓ Jobstore
✓ Queue store
```

- jf command
  - Help for all the commands
- What can you do ?
  - Get information about the project(s)
  - Get information about the jobs or flows
  - Create/manage projects
  - Deal with the runner daemon

JOBFLOW REMOTE: COMMAND LINE INTERFACE

```
(atomate2_mlff) david@mtgx-23-007:~$ jf job list
The selected project is czts from config file /home/david/.jfremote/czts.yaml
                                Jobs info
```

DB id	Name	State	Job id (Index)	Worker	Last updated [CET]
2	add	WAITING	f0951c82-e952-412… (1)	local	2024-07-02 10:30
1	add	READY	5fa275d0-6822-4a4… (1)	local	2024-07-02 10:30

```
(atomate2_mlff) david@mtgx-23-007:~$ jf flow list
The selected project is czts from config file /home/david/.jfremote/czts.yaml
                               Flows info
```

DB id	Name	State	Flow id	Num Jobs	Last updated [CET]
1	Flow	READY	8855f30e-b810-495…	2	2024-07-02 10:30

- jf command
  - Help for all the commands
- What can you do ?
  - Get information about the project(s)
  - Get information about the jobs or flows
  - Create/manage projects
  - Deal with the runner daemon

JOBFLOW REMOTE: COMMAND LINE INTERFACE

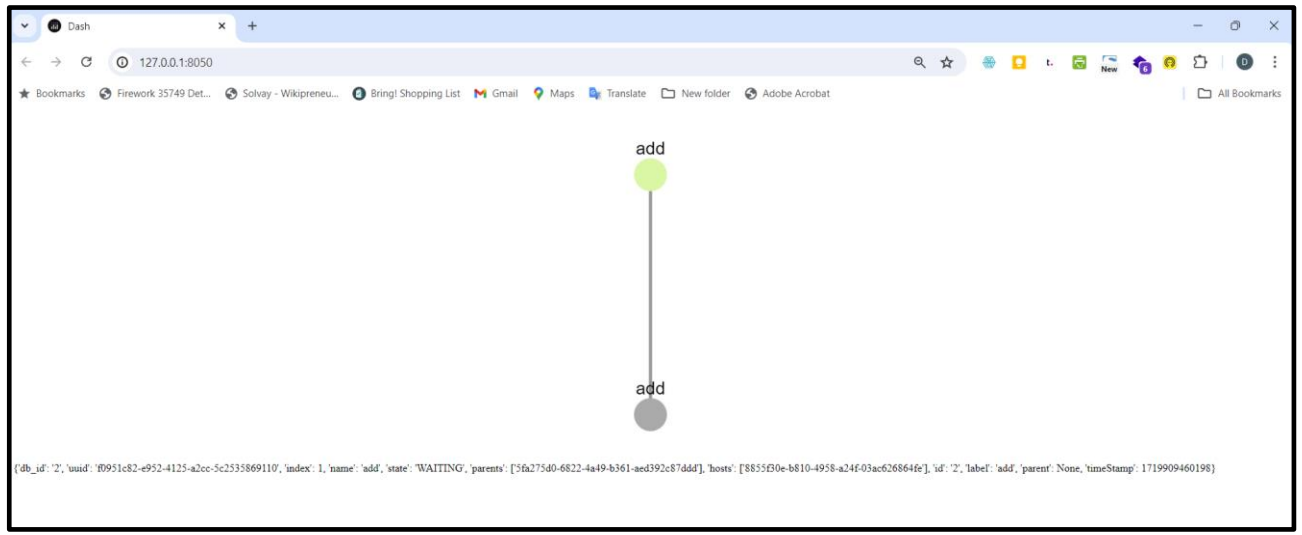
```
(atomate2_mlff) david@mtgx-23-007:~$ jf job info 1
The selected project is czts from config file /home/david/.jfremote/czts.yaml
 created_on = '2024-07-02 10:30'
      db_id = '1'
      index = 1
   metadata = {}
       name = 'add'
    parents = []
   priority = 0
     remote = {'step_attempts': 0}
      state = 'READY'
 updated_on = '2024-07-02 10:30'
       uuid = '5fa275d0-6822-4a49-b361-aed392c87ddd'
     worker = 'local'
```

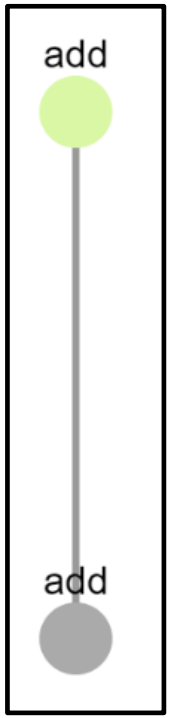
- jf command
  - Help for all the commands
- What can you do ?
  - Get information about the project(s)
  - Get information about the jobs or flows
  - Create/manage projects
  - Deal with the runner daemon

JOBFLOW REMOTE: COMMAND LINE INTERFACE

```
(atomate2_mlff) david@mtgx-23-007:~$ jf flow graph -d 1
The selected project is czts from config file /home/david/.jfremote/czts.yaml
Dash is running on http://127.0.0.1:8050/

 * Serving Flask app 'Flow - 8855f30e-b810-4958-a24f-03ac626864fe'
 * Debug mode: on
The selected project is czts from config file /home/david/.jfremote/czts.yaml
```





- jf command
  - Help for all the commands
- What can you do ?
  - Get information about the project(s)
  - Get information about the jobs or flows
  - Create/manage projects
  - Deal with the runner daemon

ADDITIONAL DETAILS

ERRORS HANDLING

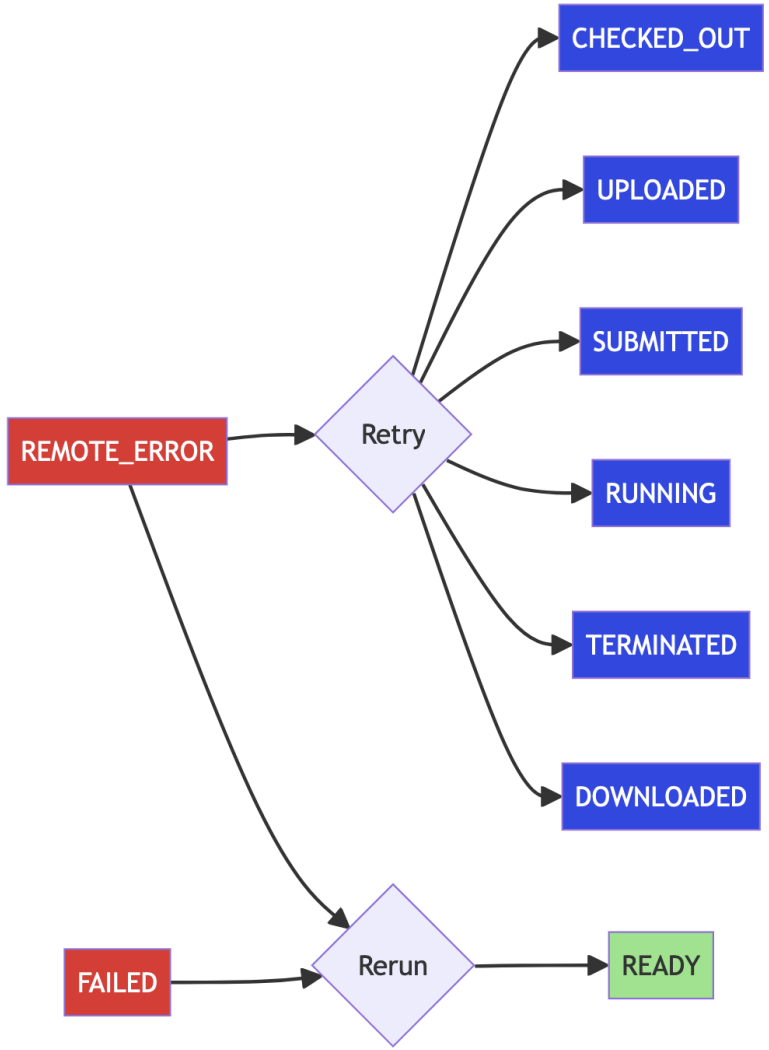
Remote errors:

- Errors during the management of the job (e.g. file transfer, queue manager,…)
- Retry: restore the Job to the state before the failure in case of temporary issues

Failed Job:

- Errors in the execution of the Job

Rerun: brings back the Job to the READY state

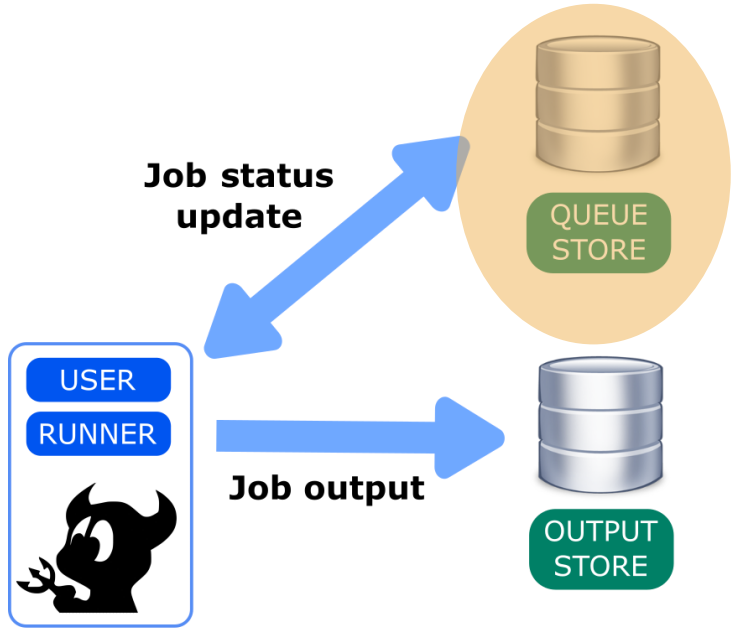


POSSIBLE LIMITATIONS

- Need to dump and transfer the output document
  - Could be a problem for large outputs
- Throughput could be limited by the daemon processes
  - Some parallelization possible in the daemon
- Set up multiple copies of the python environment in the split configurations
- Requires a Mongodb database for the queue
  - Direct access to the DB (a maggma store is not enough)

QUEUE DB STRUCTURE

- Job documents collection
- Flow documents collection
- Auxiliary collection (unique index, …)



JobDoc

- Job as_dict
- Uuid
- Index
- Db_id: unique id
- State
- Parents (uuid)
- Errors
- Run info (remote, dates, resources,…)

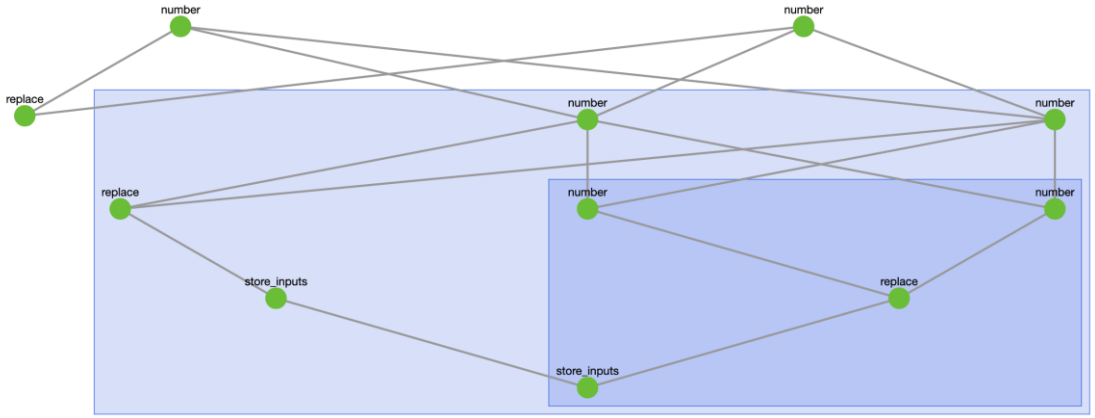
FlowDoc

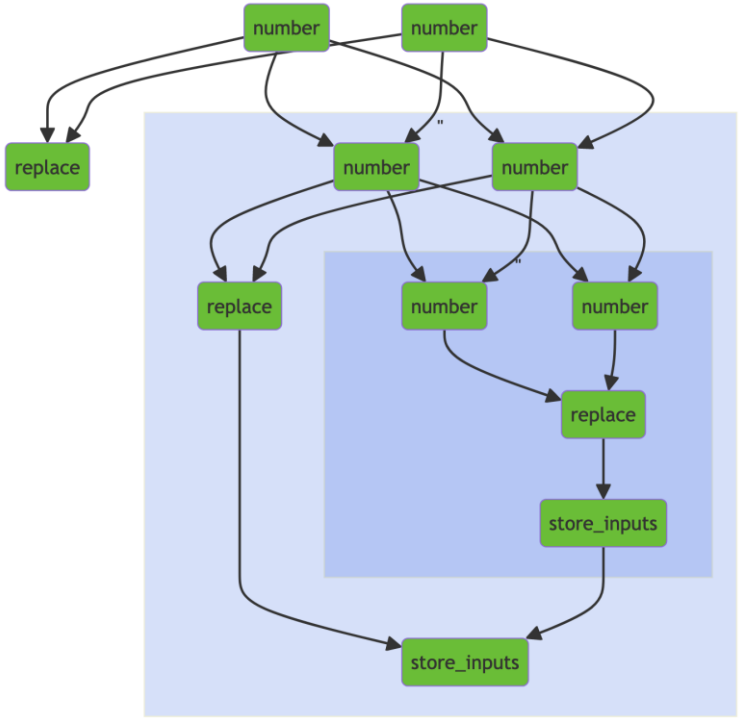
- Flow uuid
- Jobs ids (uuid, db_id, index)
- Job connections
- State

FLOW GRAPH

Minimal functionality show the graph of the flow from CLI (dash, mermaid)

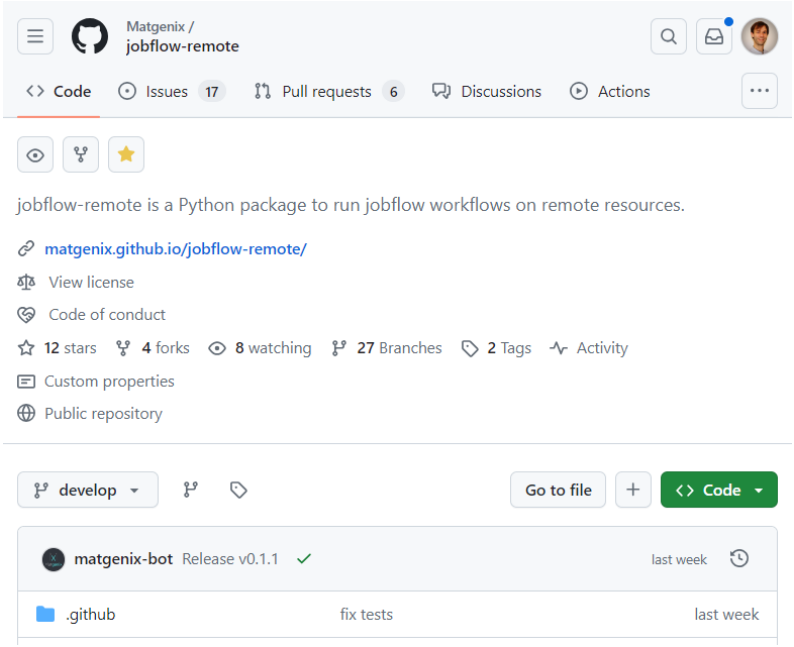
jf flow graph

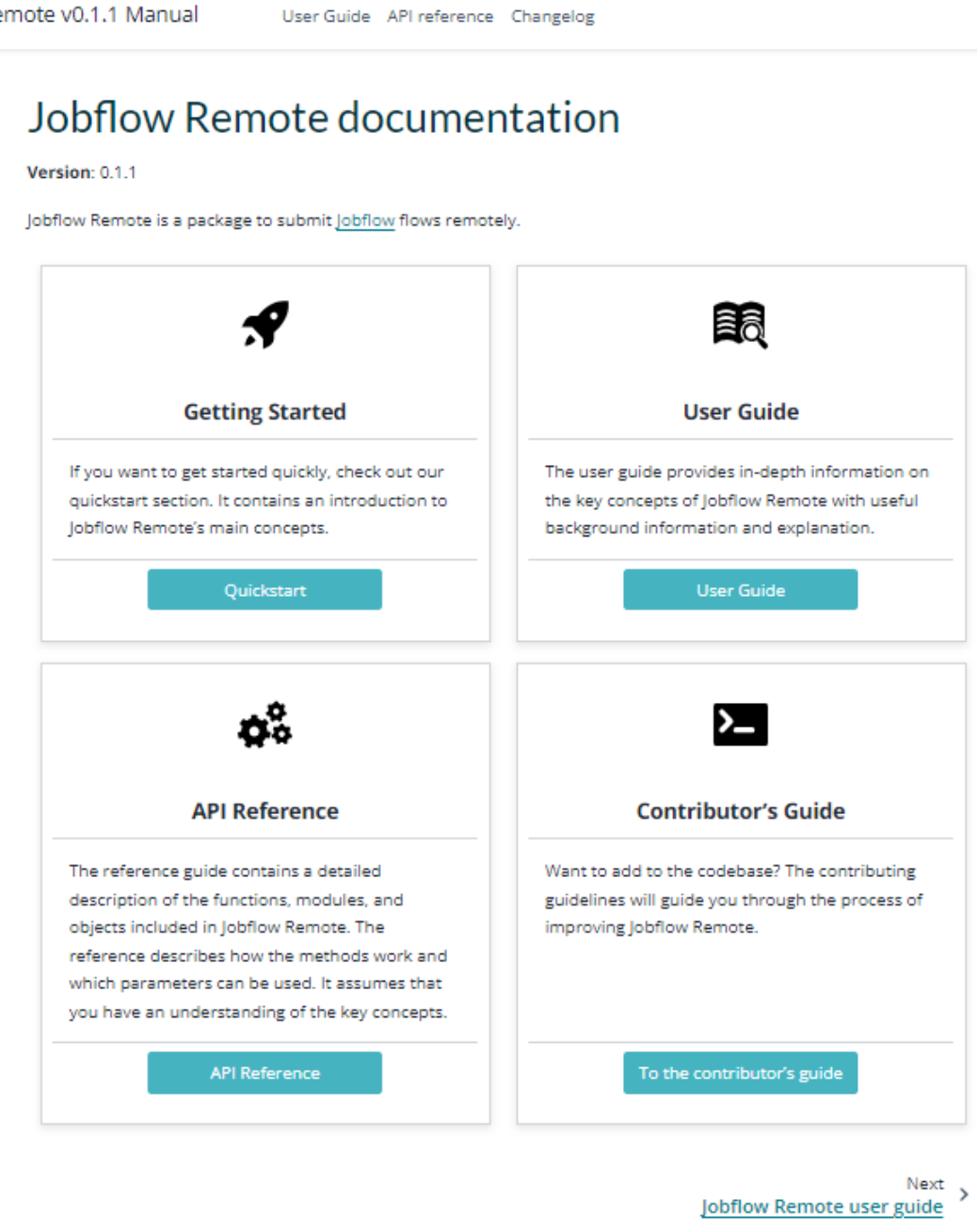




DOCUMENTATION

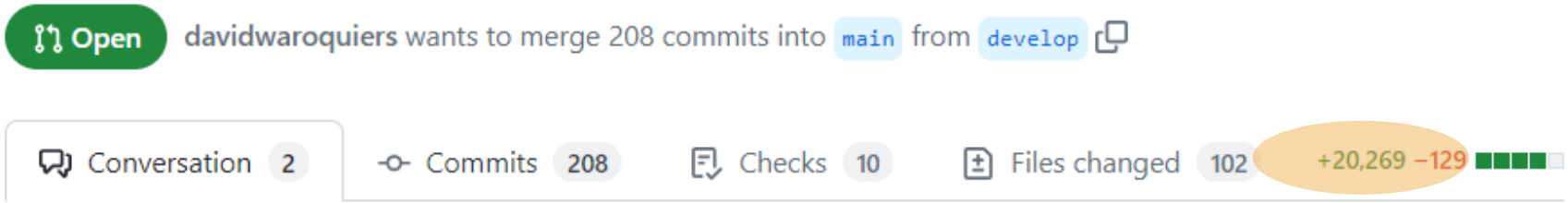
- Large online documentation:
  - https://matgenix.github.io/jobflow-remote/
- Github repository
  - https://github.com/Matgenix/jobflow-remote



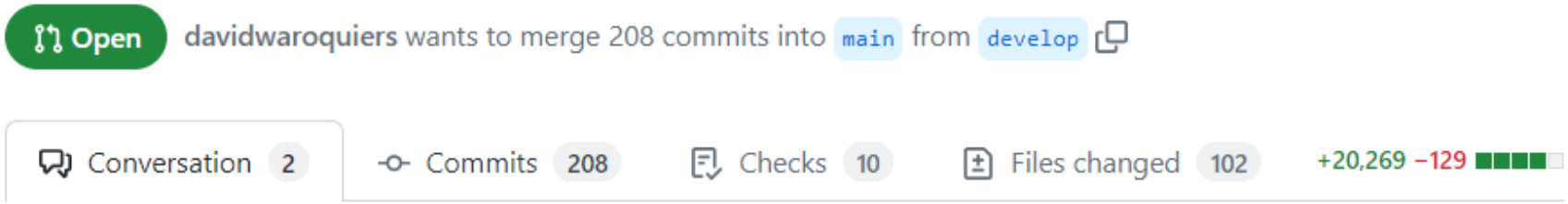


THANKS

- Guido Petretto
  - Main developer of the package

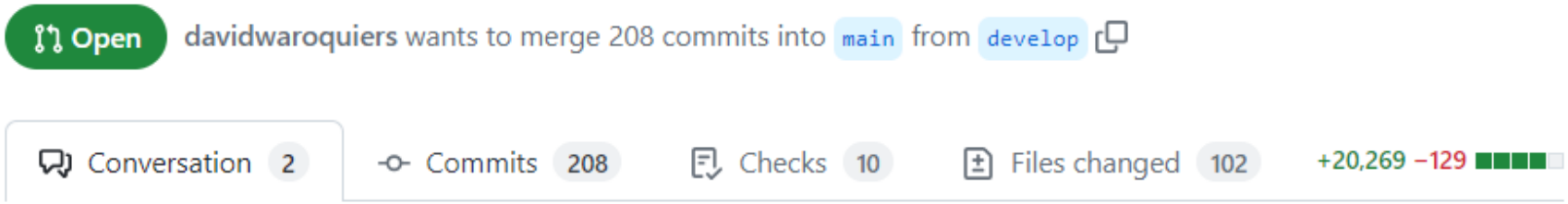


THANKS



- Guido Petretto
  - Main developer of the package
- Matthew Evans
  - Set up of CI, automated releases, docker for testing, …

THANKS



- Guido Petretto
  - Main developer of the package
- Matthew Evans
  - Set up of CI, automated releases, docker for testing, …
- Alex Ganose
  - For all the invaluable discussions to make this happen

- Anubhav Jain, Janine George, Aakash Naik, Andrew Rosen, Guillaume Brunin, Fabian Peschel, Jimmy Shen, Francesco Ricci, … and many others
  - For feedback, comments, questions, criticism, testing, …
- Umicore
  - For supporting this development and providing feedback on usage

JOB SUBMISSION: QTOOLKIT

REMOTE EXECUTION: JOB SUBMISSION

- Jobs cannot be executed directly
  - HPC infrastructure is shared between users
- A Distributed Resource Management (DRM) system (e.g. SLURM) is used to schedule jobs submitted by the users using special scripts:

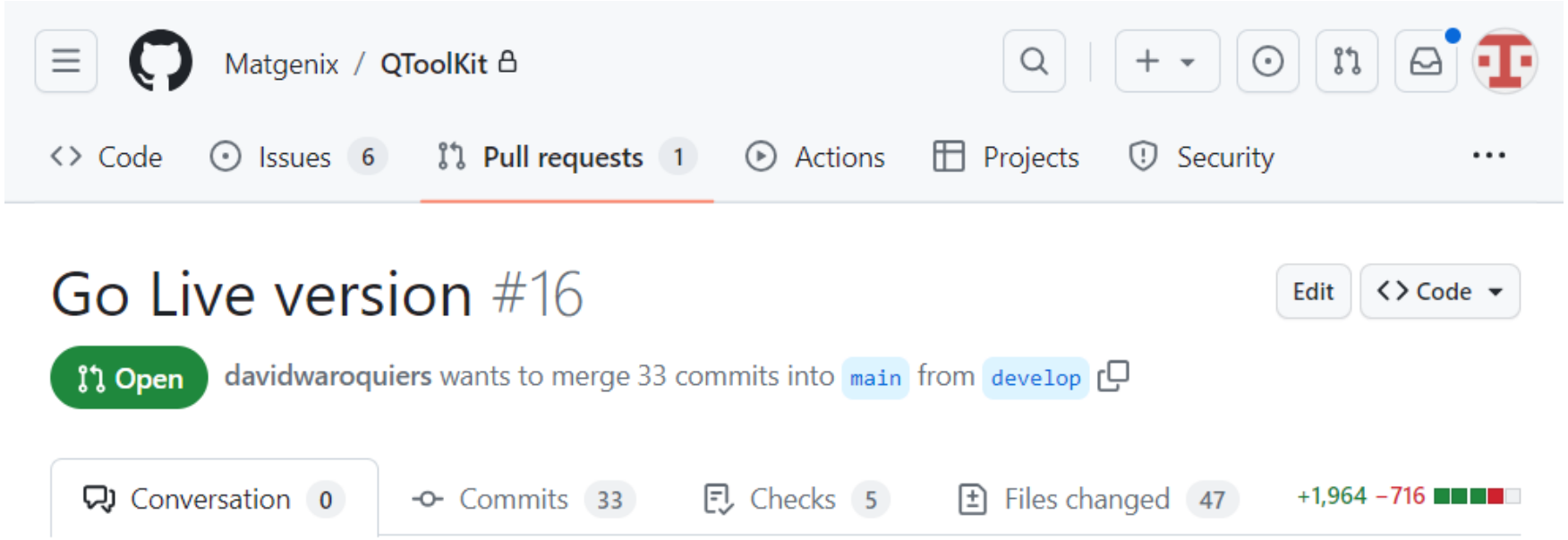

```
sbatch script.sh

#script content:
…
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=10
#SBATCH --mem=64gb
#SBATCH --time=2-00:00:00
…
```

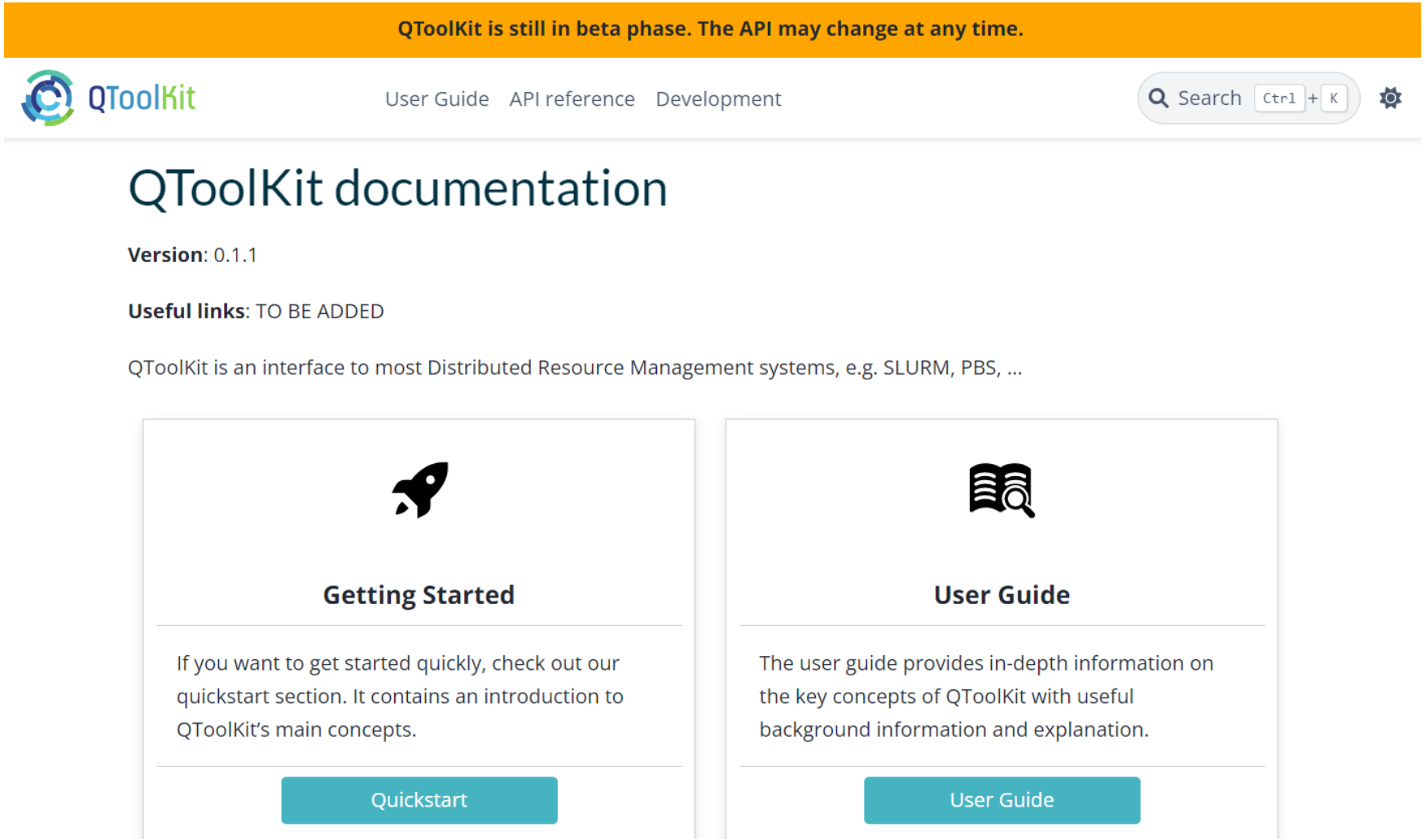
- Need for a python interface to automatically generate these scripts
- Existing solutions “buried” inside large codes or not directly usable

=> Implementation of a Python interface for jobs submission

JOB SUBMISSION: QTOOLKIT



QTOOLKIT: DOCUMENTATION



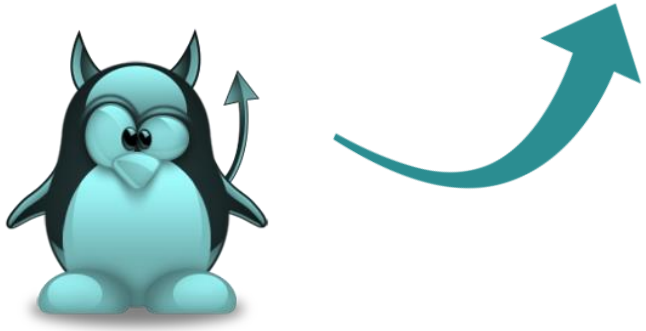
QTOOLKIT: FEATURES

- Programmatic API
- Well-defined objects to represent a job in a queue, its state, additional information, …
- Submit jobs to PBS, Slurm, Shell, …
- Get info about a job in a queue
- Get list of jobs in a queue
- No dependency on any external package (only optional dependency on monty)
- Used by jobflow-remote

REMOTE EXECUTION OVERVIEW

JOBFLOW REMOTE PACKAGE

- Runner daemon on a Work Station (WS) or Virtual Machine (VM) or “central machine” that deals with all the tasks/steps to be performed for each job
  - Runs in the background
  - Continues to run even if you exit your terminal



1. Checkout

   Get a ready job from the database
2. Upload input files

   Resolve arguments and upload files to remote HPC
3. Submit

   Job is submitted to the queue of the remote HPC
4. Check run status

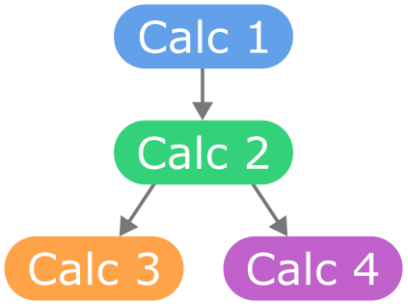
   Regularly check status in the queue
5. Download output files

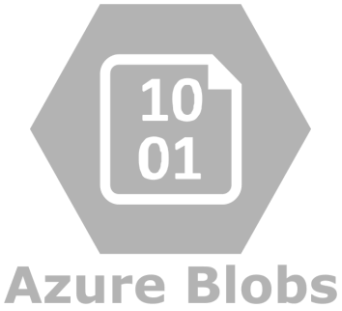
   When finished, download all outputs to WS/VM
6. Finalize

   Insert results/data in the database

JOBFLOW REMOTE PACKAGE

Insert flow

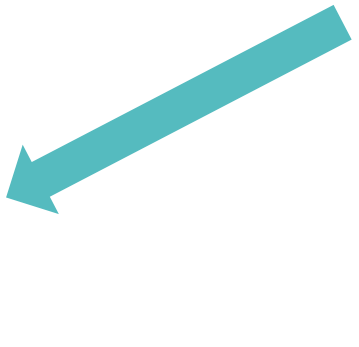




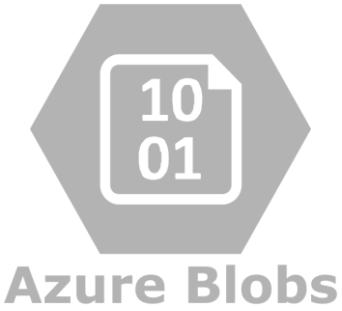
JOBFLOW REMOTE PACKAGE

Checkout job

Get a ready job from the database



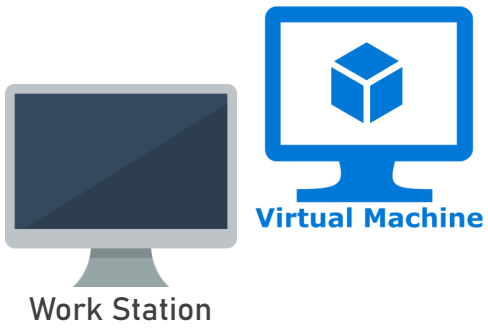
Job

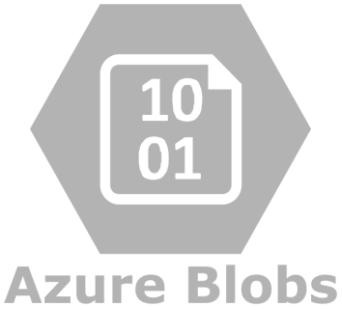


JOBFLOW REMOTE PACKAGE

Upload files

Resolve arguments and upload files to remote HPC

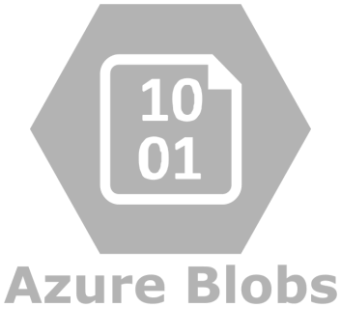




JOBFLOW REMOTE PACKAGE

Submit to HPC

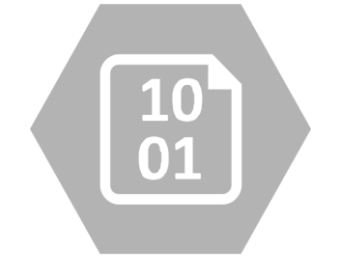
Job is submitted to the queue of the remote HPC



JOBFLOW REMOTE PACKAGE

Check status

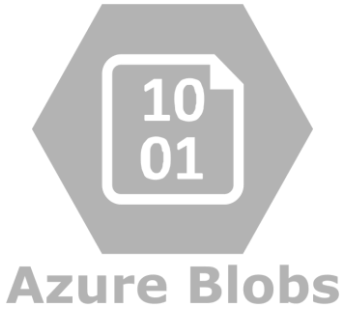
Regularly check
status in the queue



JOBFLOW REMOTE PACKAGE

Download files

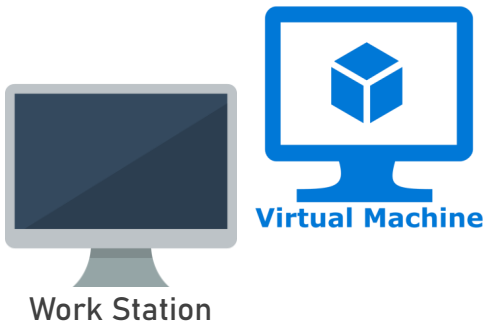
When finished,
download all outputs
to WS/VM

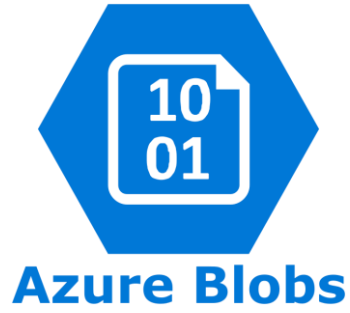


JOBFLOW REMOTE PACKAGE

Finalize

Insert results/data in the database





JOBFLOW REMOTE PACKAGE

Analyze results

Explore and analyze the results