Engaging

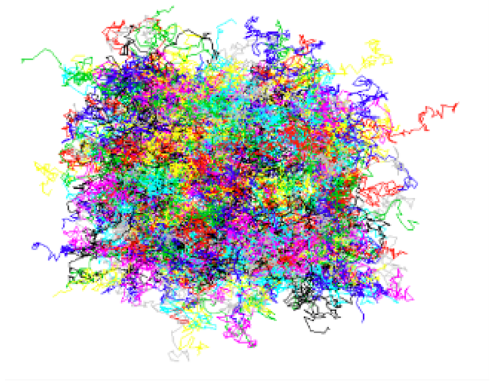
Adapted from slides by Arthur Delarue

Learning Objectives – Engaging

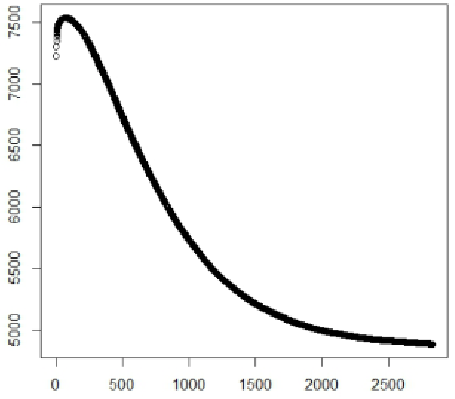
At the end of the session, students will be able to identify when to use distributed and remote computing resources and state the process for running batch and interactive jobs on Engaging.

Motivation – Distributed Computing

In optimization and stats, we often need a lot of computational power:



Machine learning on a large dataset



Simulations that require many iterations to converge

GUROBI OPTIMIZATION

Hard optimization problems

Available Resources

If you need more computational power, there are several ways you can access a remote computer or cluster:

- Another (bigger) personal computer
- Athena (at MIT)
- Resources of your department/lab (Engaging at Sloan)
- Cloud computing service (Amazon AWS, Microsoft Azure, Google Cloud Computing, etc.)

Types of Computing

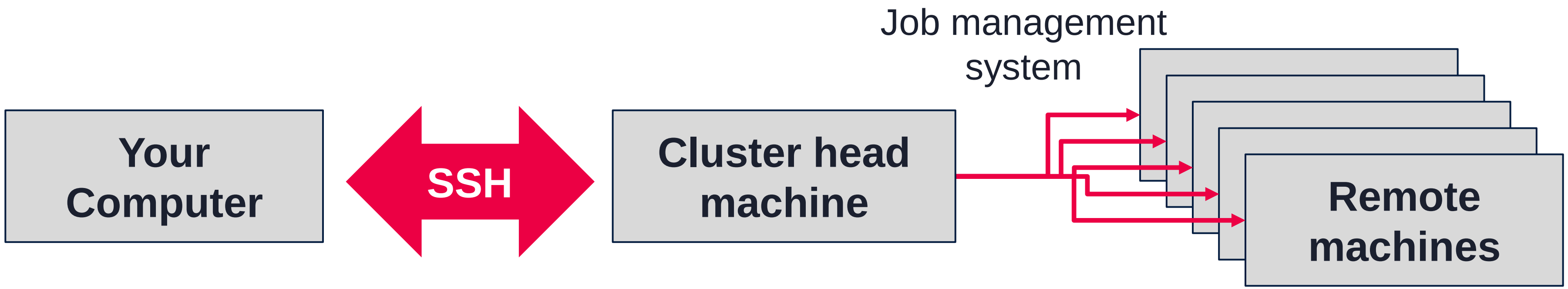
Remote computing: execute commands on a remote machine

Parallel computing: execute multiple commands simultaneously

Distributed computing: parallel computing without shared memory

Today: distributed computing with Engaging

Using a Cluster



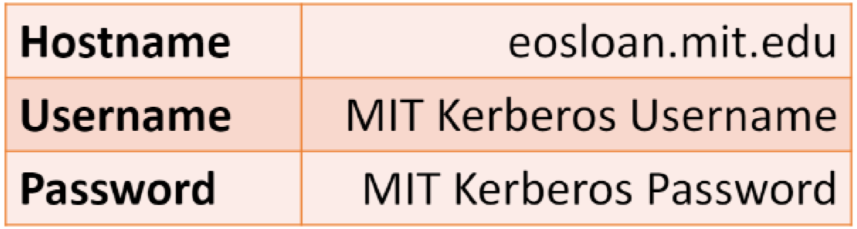
When a computing cluster is available, we can run multiple “jobs” thanks to a job management system.

- The job management system used by the Engaging cluster is called **Slurm**
- Slurm is more complex to use, but far more powerful

How to Access Engaging

1. Sloan affiliates, request an account by emailing `stshelp@mit.edu`
2. Mac: In terminal, run `ssh <username>@eosloan.mit.edu`

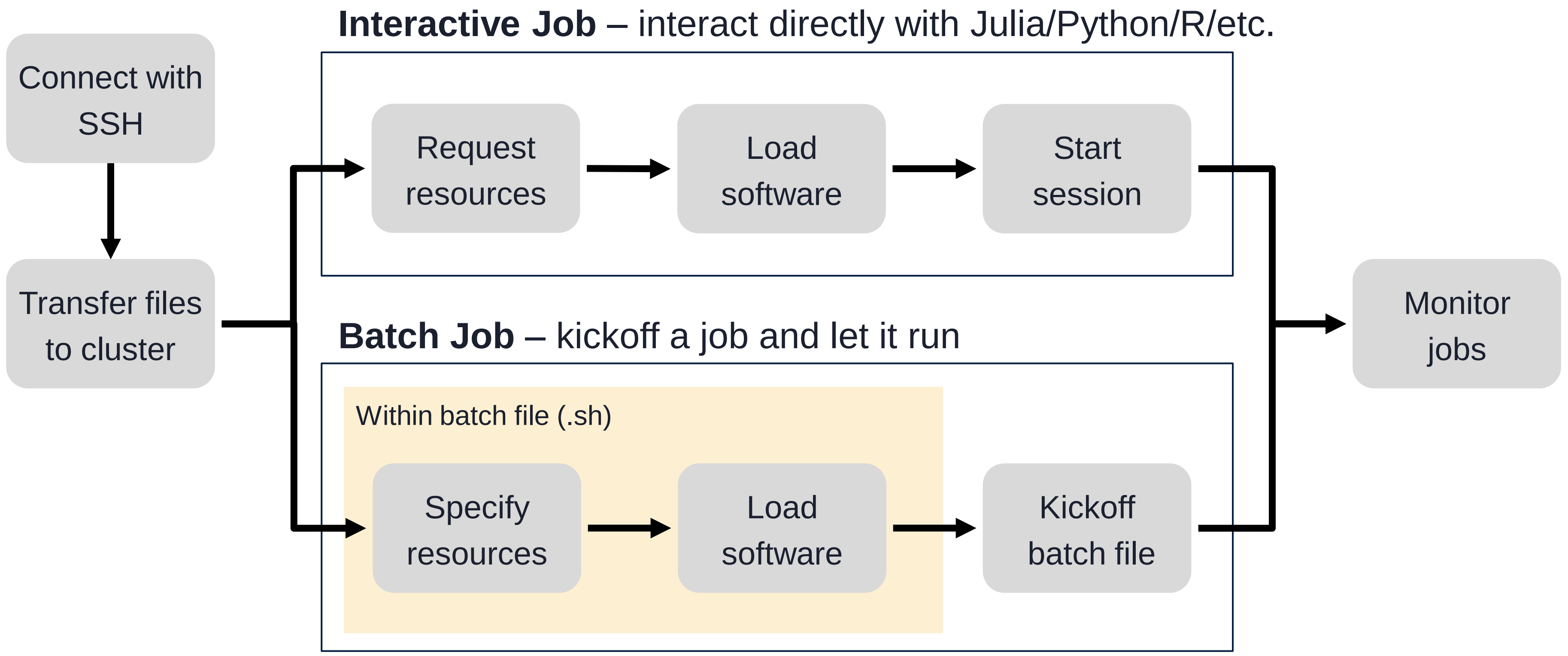
   Windows: Open PuTTY, enter host name: `eosloan.mit.edu`
3. Authenticate with username and password:



Hostname	eosloan.mit.edu
Username	MIT Kerberos Username
Password	MIT Kerberos Password

After logging in, we are on a special **login node**. The actual work will be done by the computing nodes.

Workflow



Interactive Job – interact directly with Julia/Python/R/etc.
Connect with SSH
Transfer files to cluster
Request resources
Load software
Start session
Monitor jobs
Batch Job – kickoff a job and let it run
Within batch file (.sh)
Specify resources
Load software
Kickoff batch file

File Transfer

Mac:

- Downloading from the internet (useful to install software or download datasets):

```
wget
```

- Transferring files between the local and remote machines: scp (secure copy)

```
scp file.csv <username>@eosloan.mit.edu:file.csv
```

Windows:

- Use a file transfer interface, like FileZilla or SecureFX
- I’ve used MobaXTerm which is an SSH login client (replacing PuTTY) and has a secure file transfer window

Loading Software

A variety of software is installed on the cluster, including many versions of Python, Julia, R, etc. To use them, we must load the appropriate module.

• Load a module:	`module load julia/1.5.2`
• Show all modules:	`module avail`
• Search for modules:	`eo-module-find julia`

Resources

When submitting a job, you must request the proper resources.

Task	Syntax
CPUs	`--cpus-per-task=1`
Memory	`--mem=4G`
Time	`--time=1-00:00`
Partition	`--partition=sched_mit_sloan_interactive`

Max memory usage: `1000GB`

(`mem` x `cpus-per-task`)

When the cluster is busy, your job will be queued until the resources are available. The job will fail if you use more than the allotted memory.

Partitions available (check with `eo-show-partition`):

`sched_any_quicktest` (Time limit=`0-15:00`)

`sched_mit_sloan_batch` (Time limit=`4-00:00`)

`sched_mit_sloan_interactive` (Time limit=`14-00:00`)

Your job will be queued indefinitely if your requested time is over the limit.

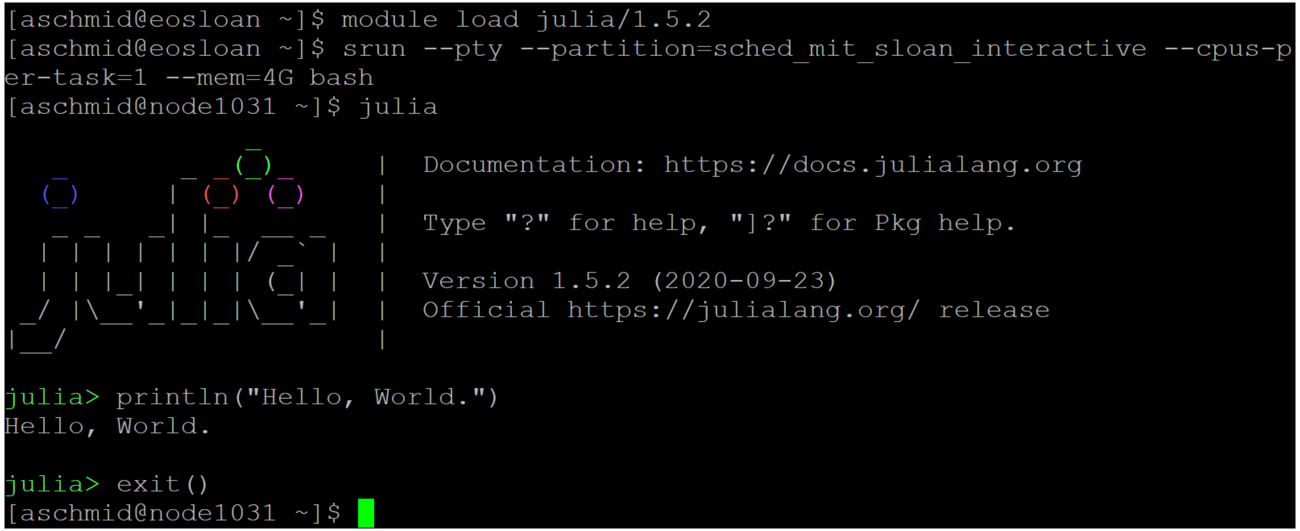
Run an Interactive Job

Load software, for example: `module load julia/1.5.2`

Request resources for an interactive job:

```
srun --pty --partition=sched_mit_sloan_interactive --cpus-per-task=1 --mem=4G bash
```

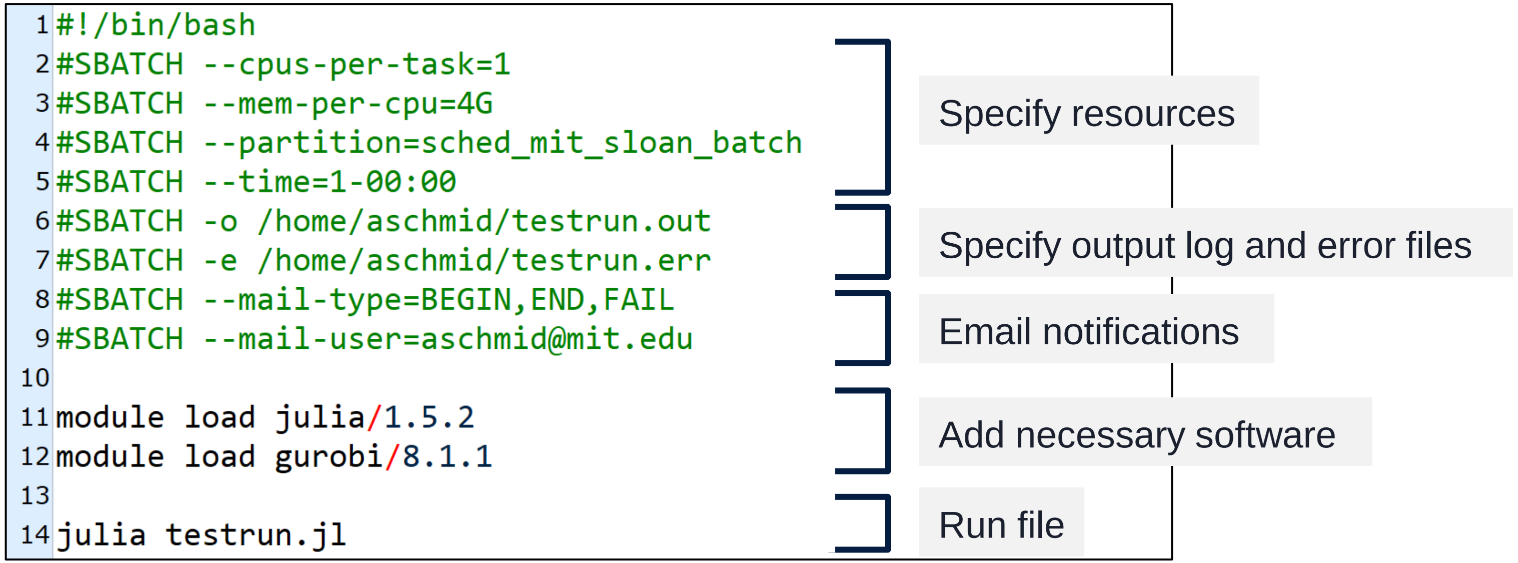
Start program, for example: `julia`



```
[aschmid@eosloan ~]$ module load julia/1.5.2
[aschmid@eosloan ~]$ srun --pty --partition=sched_mit_sloan_interactive --cpus-p
er-task=1 --mem=4G bash
[aschmid@node1031 ~]$ julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.5.2 (2020-09-23)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia> println("Hello, World.")
Hello, World.

julia> exit()
[aschmid@node1031 ~]$
```

Run a Batch Job

Create a shell script (.sh file) for the job on the cluster. For example:



```
#!/bin/bash
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=4G
#SBATCH --partition=sched_mit_sloan_batch
#SBATCH --time=1-00:00
#SBATCH -o /home/aschmid/testrun.out
#SBATCH -e /home/aschmid/testrun.err
#SBATCH --mail-type=BEGIN,END,FAIL
#SBATCH --mail-user=aschmid@mit.edu

module load julia/1.5.2
module load gurobi/8.1.1

julia testrun.jl
```

Run the batch file to kickoff the job: `sbatch filename.sh`

Never run programs on the login node!

It has very little computing power and you will paralyze it for other users submitting their jobs.

Always run interactive jobs with `srun` and batch jobs with `sbatch`, otherwise your job will run on the login node.

Monitoring Jobs

It’s often useful to track the progress of your jobs and see how busy the cluster is.

- See your submitted jobs: `eo-show-myjobs`
- See all submitted jobs: `eo-show-alljobs`
- Cancel a job: `scancel <JOBID>`

(find `JOBID` using `eo-show-myjobs`)

Poll Question

Which of these commands would kick off an interactive session properly?

A)
```
srun --partition=sched_mit_sloan_interactive --cpus-per-task=1 --mem=4G bash
```

B)
```
--partition=sched_mit_sloan_interactive --cpus-per-task=1 --mem=4G bash
```

C)
```
srun --pty --partition=sched_mit_sloan_interactive bash
```

D)
```
srun --pty --partition=sched_mit_sloan_interactive --cpus-per-task=1 --mem=4G bash
```

E) Both C and D

Poll Question

Which of these commands would kick off an interactive session properly?

A) `srun --partition=sched_mit_sloan_interactive --cpus-per-task=1 --mem=4G bash`

B) `--partition=sched_mit_sloan_interactive --cpus-per-task=1 --mem=4G bash`

C) `srun --pty --partition=sched_mit_sloan_interactive bash`

D) `srun --pty --partition=sched_mit_sloan_interactive --cpus-per-task=1 --mem=4G bash`

[✓ **E)** Both C and D]

Running jobs with many parameters

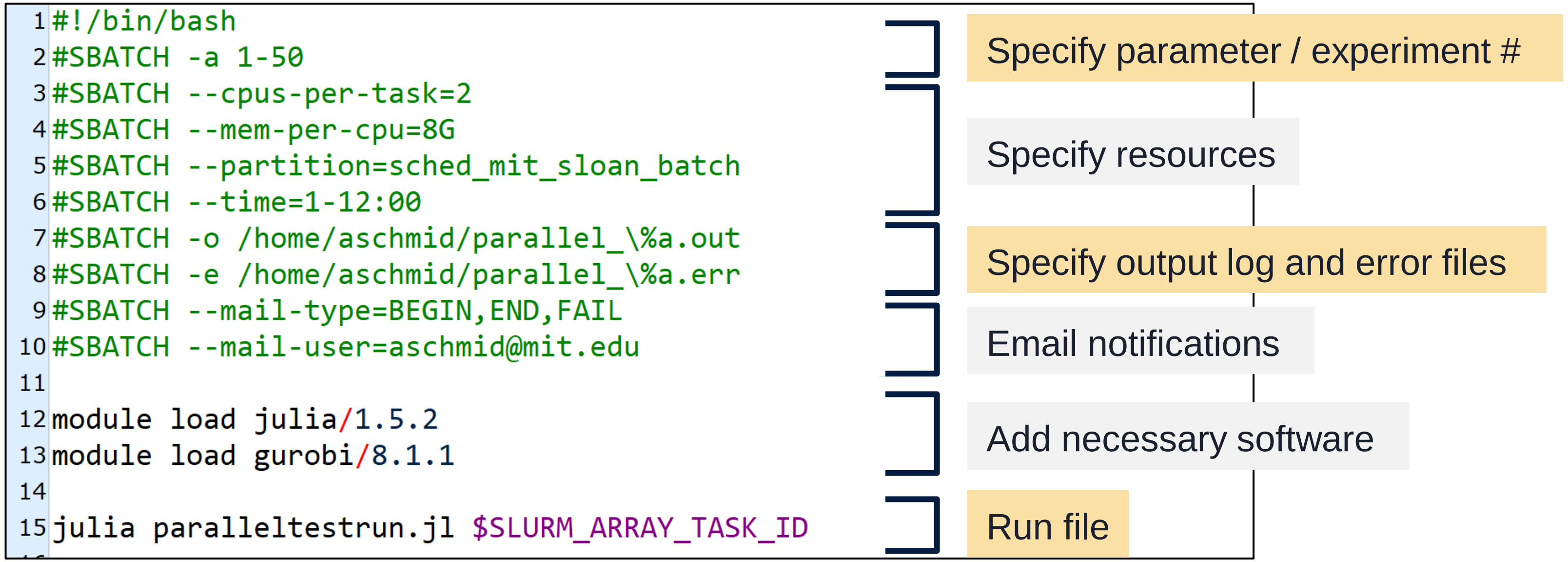
We often want to run the same code with different sets of parameters (e.g. machine learning parameter tuning, running several instances of an optimization problem)

Option 1: We can create a file for each parameter we want to test, modify the parameter in each file, create a batch file for each, then kick off each separately ☠️

Option 2: We can kickoff a single parallel batch file, triggering jobs with each different parameter to generate at once ✅

Parallel Jobs

Create a .sh file for the array of jobs. For example:



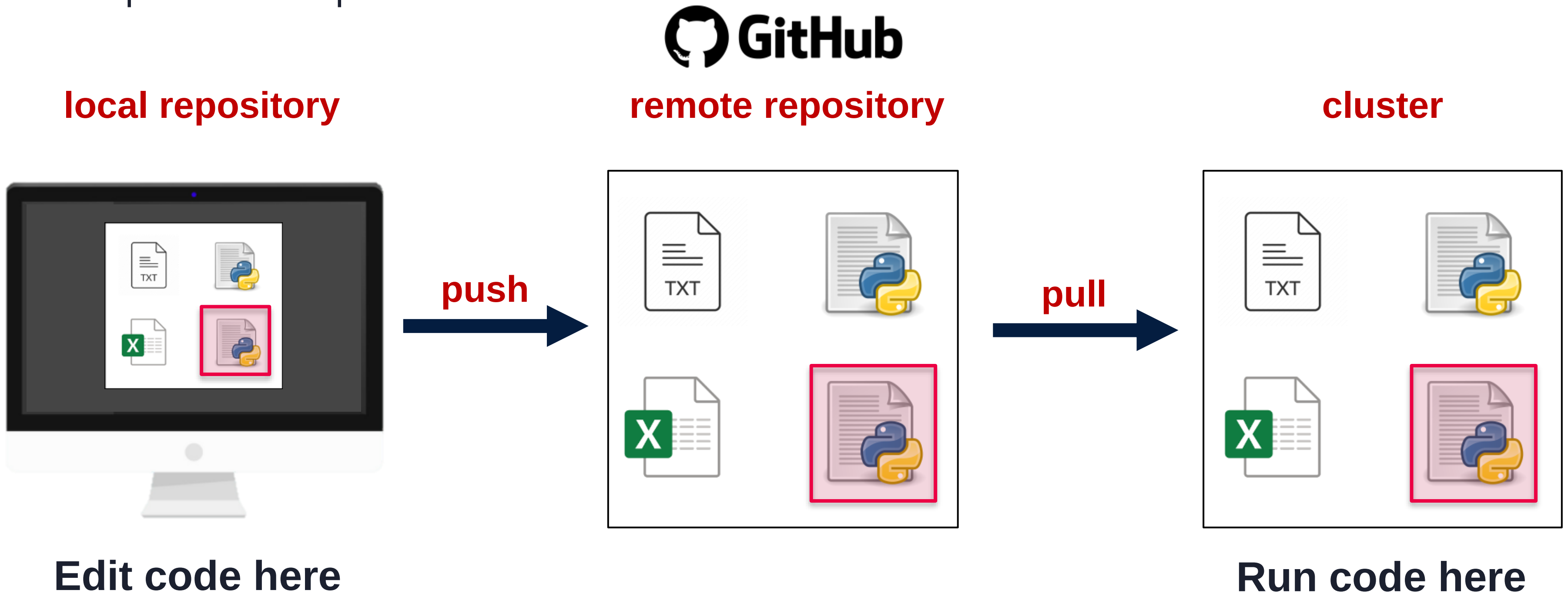
```
#!/bin/bash
#SBATCH -a 1-50
#SBATCH --cpus-per-task=2
#SBATCH --mem-per-cpu=8G
#SBATCH --partition=sched_mit_sloan_batch
#SBATCH --time=1-12:00
#SBATCH -o /home/aschmid/parallel_\%a.out
#SBATCH -e /home/aschmid/parallel_\%a.err
#SBATCH --mail-type=BEGIN,END,FAIL
#SBATCH --mail-user=aschmid@mit.edu

module load julia/1.5.2
module load gurobi/8.1.1

julia paralleltestrun.jl $SLURM_ARRAY_TASK_ID
```

Specify parameter / experiment #
Specify resources
Specify output log and error files
Email notifications
Add necessary software
Run file

Run the batch file to kickoff the array of jobs: `sbatch filename.sh`

Workflow for Github + Engaging

Connect the cluster to your Github, then clone and pull from your repos to run computational experiments



Connecting Engaging to Github

1. Follow the link below for instructions on how to generate an SSH key for your Github account, add your key to the SSH agent, then add the key to your Github account. https://docs.github.com/en/get-started/quickstart/set-up-git

2. Clone the Github repository from the cluster, i.e. after you have connected via SSH

```
git clone https://github.com/[username]/[reponame].git
```

3. As Github is updated, pull from the repository using same syntax as usual

```
git pull
```

The files are now on the cluster and you can run jobs as you would with any other files!

Key Takeaways

- Use the cluster to boost your productivity and save your own computing resources
  - Interactive job – code in your favorite language interactively
  - Batch job – kickoff and check back when it finishes
- Check out parallel batch jobs to efficiently test different parameters
- Connect to Engaging to Github to easily keep track of code changes and run computational experiments (try it out in the homework)
- Revisit this section later in the class when you have more tools. Especially session 8!

Do not run jobs on the login node!! Use `srun` **or** `sbatch`

More Resources

- Check out other clusters available:
  - SuperCloud (requires access, GPUs available, https://supercloud.mit.edu/)
  - Satori (instant access to MIT students, https://mit-satori.github.io/satori-ssh.html)
- Further documentation for Engaging available at: https://wikis.mit.edu/confluence/display/sloanrc/Engaging+Platform
  - More examples and options for batch and interactive jobs
  - Available storage and resources

Break

Resume in 10 minutes