Running jobs on the Peregrine computer cluster

https://github.com/richelbilderbeek/Peregrine

What, Why, Mastery

- I will show how to set up one to many jobs (without dependencies)
- Starting each job manually is tedius
- You can create one to many jobs

- Not: running jobs that are dependent on another

Overview

- bash #1
- Job without replicates
- bash #2
- Job with replicates
- bash #3
- Conclusion

bash #1

```
echo "Hello world"
```

Display ‘Hello World’

```
ls
```

List directory content

```
echo "Hello world" > tmp.txt
```

Streams the text ‘Hello world’ to file tmp.txt in overwrite mode

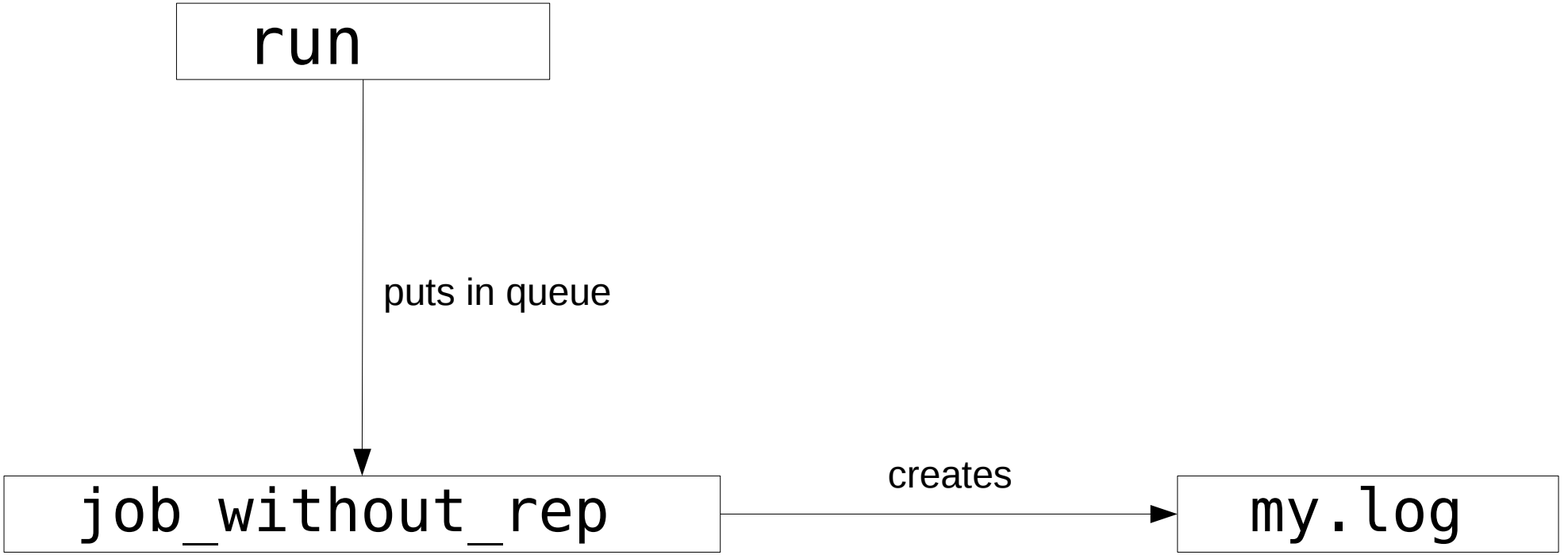
```
cat tmp.txt
```

Shows the file tmp.txt

```
./run.sh
```

Run the local executable bash script called run.sh

Job without replicates



Run and check it

```
p230198@pg-login:job_without_rep ./run
Submitted batch job 12345678
```

```
p230198@pg-login:job_without_rep cat my.log
241
```

run

```
#!/bin/bash
sbatch job_without_rep
```

job_without_rep

```
#!/bin/bash
#SBATCH --time=0:00:01
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --ntasks=1
#SBATCH --mem=1M
#SBATCH --job-name=job_without_rep
#SBATCH --output=my.log
echo $RANDOM
```

bash #2

```
echo "Hello world" | cut -d " " -f 1
```

Pipe the words ‘Hello world’ to cut. Show, using space as a delimiter, the first field

```
echo "Hello world" | cut -d " " -f 1 > tmp.txt
```

Pipe the words ‘Hello world’ to cut.
Stream, using space as a delimiter, the first field to tmp.txt in overwrite mode

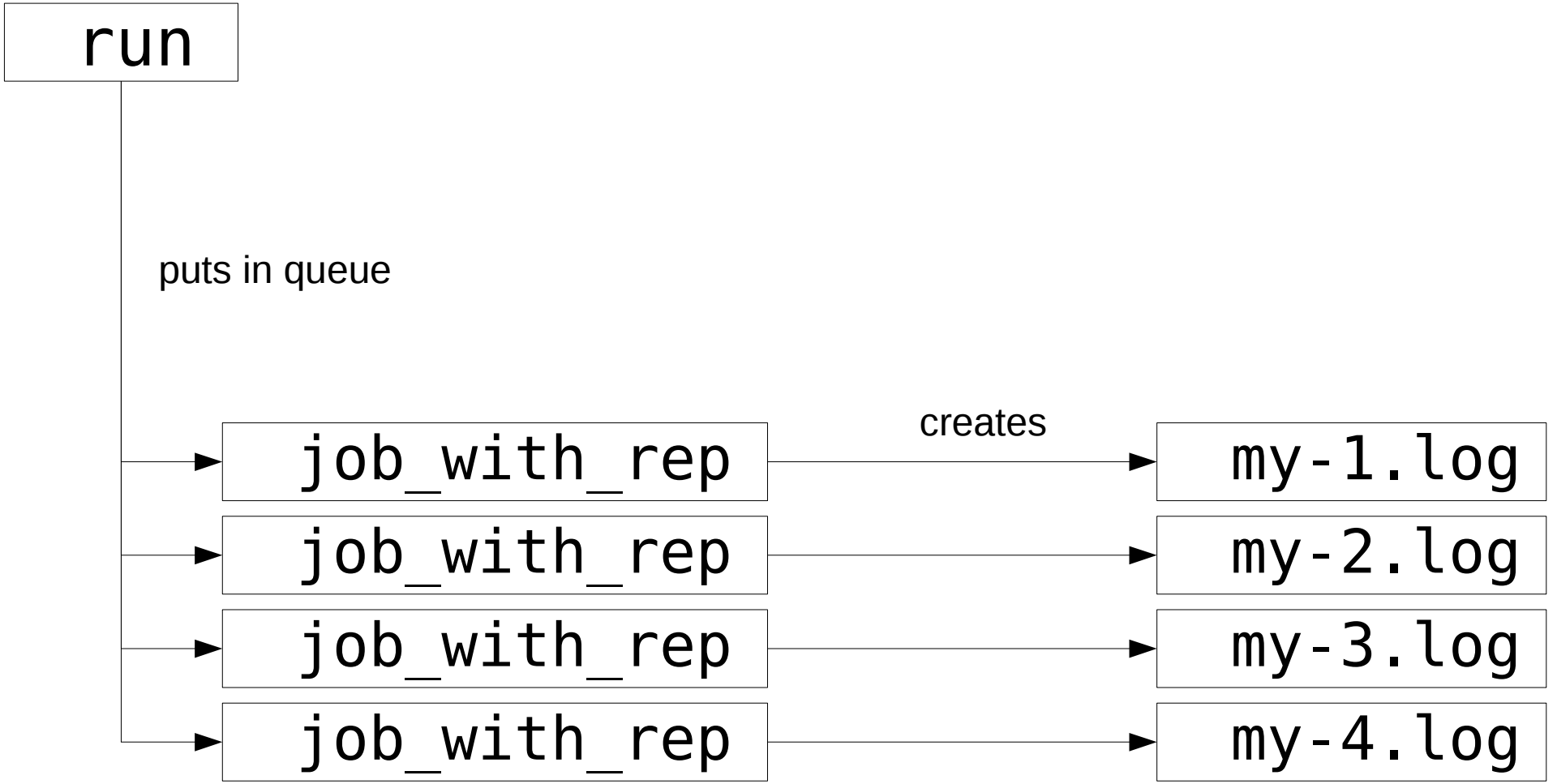
```
echo "Hello world" | cut -d " " -f 2 >> tmp.txt
```

Pipe the words ‘Hello world’ to cut.
Append, using space as a delimiter, the second field to tmp.txt

```
hi=`echo "Hello world" | cut -d " " -f 1`
echo $hi
```

Pipe the words ‘Hello world’ to cut.
Extract, using space as a delimiter, the first field.
Store this result in the variable called ‘hi’
Display the content of the variable ‘hi’

Job with replicates



Run and check it

```
p230198@pg-login:job_with_rep ./run
Submitted batch job 12345678
```

```
p230198@pg-login:job_with_rep cat *.log
1234
4321
1111
42
```

run

```
#!/bin/bash
for i in {1..4}
do
  sbatch job_with_rep
done
```

job_with_rep

```
#!/bin/bash
#SBATCH --time=0:00:01
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --ntasks=1
#SBATCH --mem=1M
#SBATCH --job-name=job_with_rep
#SBATCH --output=my-%j.log
echo $RANDOM
```

bash #3

```
egrep -R “error”
```

Searches all files in folders and subfolders for the regex ‘error’

```
egrep -iR “error”
```

Searches all files in folders and subfolders for the regex ‘error’, case insensitively

```
egrep -iR “error” --include=*.log
```

Searches all files with a .log file extension in folders and subfolders for the regex ‘error’, case insensitively

```
scp p230198@peregrine.hpc.rug.nl:/home/p230198/any_folder/*.*
~/any_other_folder
```

Securely copy all files from a Peregrine folder to a folder in your LWP home

```
echo “alias q='squeue -u $USER'” >> ~/.bashrc
```

Use the command ‘q’ to show your jobs in the queue (after a restart)

Conclusion

- Setting up jobs is easy
- Some knowledge of bash is useful