COMPUTATIONAL BIOLOGY LAB

Guía de usuario inicial

El clúster del CBLab

Nº nodo	Procesadores	RAM
node110	64	515
node112	20	128
node111	20	193
node109	20	193
node108	20	193
node107	20	193
node106	20	193
node105	20	193
node104	16	161
node103	16	161
node102	16	161
node101	16	161
node100	16	161

Almacenamiento Array de discos 40Tb en RAID 5 → Diversas bases de datos, datos de los usuarios, programas instalados en /home/soft

Logincluster 1 y Logincluster 2 → Dentro está la máquina virtual cluster-ceab para el acceso al cluster

Mastercluster 1 y Mastercluster 2 → Gestión de las colas y las imágenes de los nodos

Fibercluster 1 y Fibercluster 2 → Controla el array de discos.

Login (Unix y Mac)

Abrimos una terminal o consola y escribimos:

ssh <user>@cluster-ceab.ceab.csic.es

Introducimos la contraseña (sólo tenemos tres intentos).

Si nos equivocamos dos veces usar **ctrl+C** para no bloquear la **IP**

Ahora nos encontramos en la home de nuestro usuario en cluster-ceab, la máquina virtual que hace de puerta al clúster.

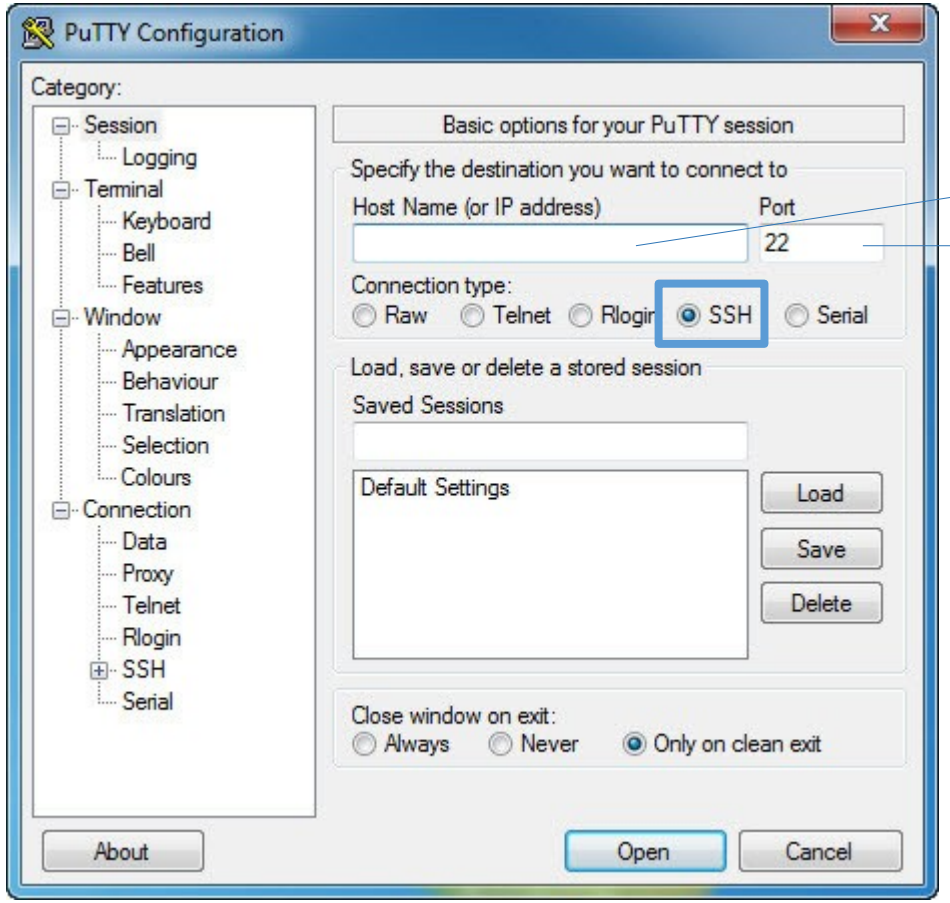
En /home/usuaris están todos los usuarios y en /home/soft encontramos los programas compilados

```
miguel@kodama:~$ ssh user@cluster-ceab.ceab.csic.es
user@cluster-ceab.ceab.csic.es's password:
Last login: Mon Mar 27 00:34:01 2017 from 87.125.67.9
user@cluster-ceab:~$



user@cluster-ceab:/home/usuaris$ ls
admins_ceab   barbosa      charlie     fbartu
jcapitan      j.palmer     mcefali     mturon
ramon         test-rdlab   vicente     afernand
calonectris   chuetestauf  flucati     j.drummond  [...]

user@cluster-ceab:/home/usuaris$ cd /home/soft
user@cluster-ceab:/home/soft$ ls
arb           fastx_toolkit-0.0.14     iprscan
mpich32       pear-0.9.6-bin-64        R-3.0.2jm
smallfile     usearch-8.1.18           arb-6.0
gatk          JAGS                     mrbayes
PgRouting     R-3.1.2                  sources
velvet_1.2.10 banjo     [...]
```

Login en cluster-ceab y muestra de /home/usuaris y /home/soft

Login (Windows)

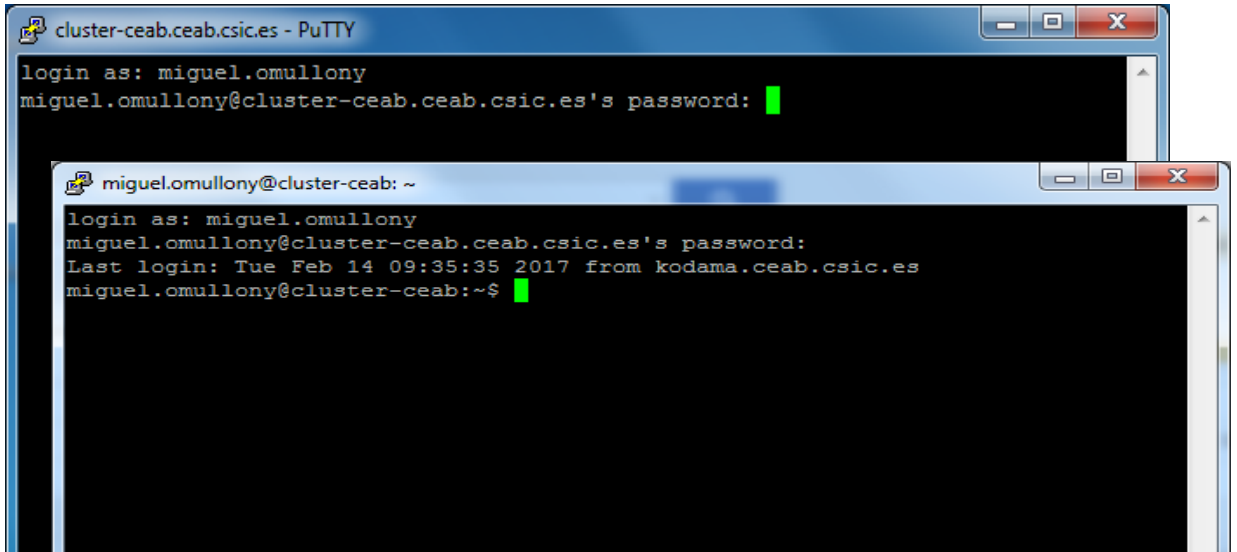
- Usaremos el programa PuTTY:



cluster-ceab.ceab.csic.es

22 (aparece por defecto)

Se abrirá una ventana donde escribiremos nuestro nombre de usuario y nos pedirá la contraseña



Comandos básicos

Para algunos será la primera vez que veáis una shell. Aquí unos comandos para ir aprendiendo y quitarse el miedo:

mkdir <nom_directorio> → se usa para crear directorios (carpetas).

cd <nom_directorio> → si existe, para entrar en un directorio. **cd ..** nos lleva a la carpeta superior y **cd** a nuestra home.

ls → lista los archivos que hay dentro de un directorio. Podemos usar diferentes opciones para obtener más información, por ej, **ls -lt** nos muestra los archivos ordenados por fecha de modificación.

Los comandos se pueden combinar con el uso del **PATH**. El path es la "dirección" o el lugar dónde están los archivos o carpetas.

```
user@cluster-ceab:~$ mkdir ej_diap

user@cluster-ceab:~$ cd ej_diap

user@cluster-ceab:~/ej_diap$ ls

user@cluster-ceab:~/ej_diap$ touch archivo.txt

user@cluster-ceab:~/ej_diap$ ls
archivo.txt

user@cluster-ceab:~/ej_diap$ cd

user@cluster-ceab:~$ mkdir ej_diap/new_dir

user@cluster-ceab:~$ ls ej_diap
archivo.txt new_dir
```

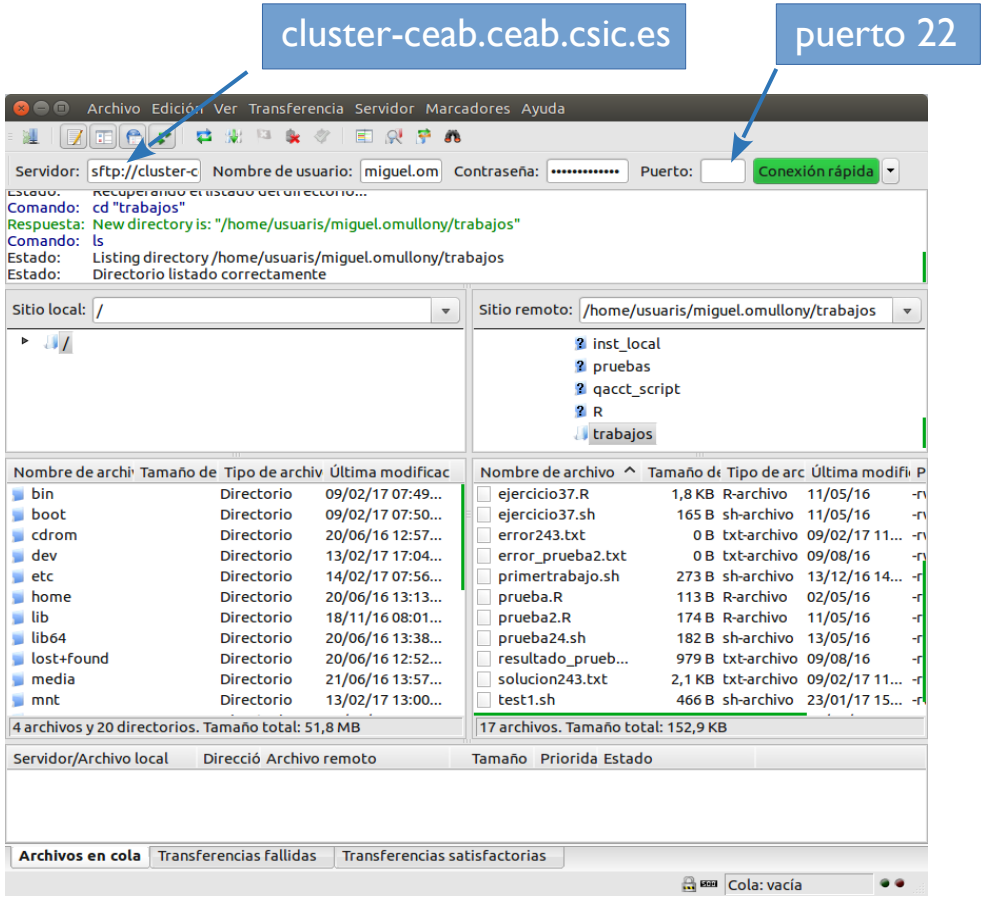
Uso de Filezilla

Con Filezilla podemos intercambiar archivos entre nuestro ordenador y el clúster.

Existe versión para Win/Mac/Linux: *Filezilla*

Se pueden arrastrar los archivos directamente a la ventana.

Si no te acostumbras a usar los comandos en la terminal puedes navegar entre los ficheros del clúster desde Filezilla



Envío de trabajos. Script

Para enviar los trabajos primero necesitamos hacer un script de bash.

El archivo debe estar en formato de texto plano, se puede usar el editor nano.

nano <nom_script>.sh

Un script básico:
En la primera línea escribimos **#!/bin/bash** para indicarle que es un script que usa bash.
En la segunda el path del directorio donde están los datos y en la tercera el path del programa con el fichero de entrada (<) salida (>) y error (2>).

Para cerrar nano usamos **ctrl+X**, nos pedirá confirmación de nombre y guardado (Y/N). Si hemos abierto nano sin indicar un nombre podremos escribirlo ahora.

```
# Aquí un ejemplo de un script sencillo:

user@cluster-ceab:$ nano test1.sh

#abre el editor nano y podemos empezar a escribir:
```

```
GNU nano 2.2.6     Archivo:test1.sh
#!/bin/bash

cd /home/usuaris/<user>/<data_folder>

/home/soft/<programa> < fichero_entrada >
output.txt 2> error.txt

# todo lo que haya detrás de un # es un
# comentario.
# el > output.txt y el 2> error.txt son opcionales
# pero útiles para tener información si un trabajo
# falla.

^G Ayuda   ^O Guardar   ^Y Pág. ant.   ^K Cortar
^X Salir   ^W Buscar    ^V Pág. sig.   ^U Pegar
```

Envío de trabajos. Script

Ejemplo de un script para un programa que requiera PATHS y librerías

En el caso de programas compilados en /home/soft hay que indicar el path del binario ejecutable:
export PATH=/home/soft/~~:$PATH

Pueden ser una o varias librerías, hay que indicarlas en cada línea con
export LD_LIBRARY_PATH=/~~:$LD_LIBRARY_PATH

```
GNU nano 2.2.6                    Archivo: test1.sh                    Modificado

#!/bin/bash

# You can write comments after #

# location of code or data that you use:

cd /home/usuaris/<user_name>/<work_directory>

### PATHS if neccesary,

export PATH=/home/soft/<program_dir>/bin:$PATH

### LIBRARIES PATHS

export LD_LIBRARY_PATH=/home/soft/<program_dir>/lib:$LD_LIBRARY_PATH

### COMMAND without line feeds(saltos de linea), even if it's too long.

/home/soft/<program_dir> <options> < file(R, python...) > output_name.txt 2> error_output_name.txt

^G Ver ayuda   ^O Guardar   ^R Leer fich.   ^Y Pág. ant.   ^K Cortar Texto   ^C Posición
^X Salir       ^J Justificar ^W Buscar      ^V Pág. sig.  ^U PegarTxt       ^T Ortografía
```

Carpeta o directorio donde se encuentra el código o los datos que vamos a ejecutar

Archivo que queremos enviar

Ficheros de salida y error (más información en caso de fallos)

Envío de trabajos. qsub

Es el comando para poner un trabajo en cola. La forma básica:

```
qsub <nom_script>.sh
```

Pero podemos ir añadiendo opciones dependiendo de los recursos que necesitemos

- Asignar X gigas de RAM

```
qsub -l h_vmem=XG <nom_script>
```

- Si queremos paralelizar dentro de un nodo escogemos Y procesadores:

```
qsub -pe make Y <nom_script>
```

```
user@cluster-ceab:$ qsub test1.sh
Your job 27323 ("test1.sh") has been submitted

user@cluster-ceab:$ qsub -l h_vmem=8G -pe make 10 test1.sh

Your job 27325 ("test1.sh") has been submitted

user@cluster-ceab:$ ls

test1.sh.e27323  test1.sh.o27325
Test1.sh.e27325  test1.sh.pe27325
test1.sh.o27323  test1.sh.po27325
```

Y como siempre las opciones se pueden combinar, teniendo en cuenta que la RAM se asigna por procesador. Un ejemplo, queremos enviar un trabajo de 5 procesadores y 100G de RAM:

```
qsub -l h_vmem=20G -pe make 5 <nom_script>.sh
```

Envío de trabajos. qsub

- Enviar trabajo a una cola(nodo) determinada:

qsub -q ceab@nodexxx

Cuidado con no superar los parámetros del nodo al que lo envías.

- Recibir notificaciones al empezar, terminar o borrar un trabajo:

qsub -m bea -M <correo@electrónico.ej>

Todas las opciones se envían en una misma línea, con el script.sh al final. Cuidado con los espacios y con pulsar "intro" antes de tiempo.

Importante: Actualmente la máxima cantidad de RAM disponible es de 515G y la paralelización se hace intra-nodo. Tenemos 16x5, 20x6 y 64 procesadores. Si lanzamos un trabajo con 20 procesadores y mem=30G podría entrar en los nodos de 20 o el de 64, pero estamos reservando 600G de RAM (20x30)... Y no funcionará.

```
user@cluster-ceab:$ qsub -l h_vmem=8G -pe make 20
-m bea -M cblab@ceab.csic.es -q ceab@node111
test_1.sh

Your job 27455 ("test_1.sh") has been submitted
```

qstat y qdel

Una vez enviado un trabajo podemos ver en qué cola(nodo) y en qué estado se encuentra con **qstat**

r corriendo, **qw** en espera, **Eqw** en error.

Otras opciones:

qstat -u "*" → trabajos de todos los usuarios

qstat -r (-u "*") → recursos que estás usando (o todos los usuarios)

qstat -j <id_trabajo> → más información sobre el trabajo

Si queremos borrar un trabajo de la cola:

qdel -j <id_trabajo>

```
user@cluster-ceab:~/Structure_pruebas$ qstat
job-ID  prior   name       user         state submit/ start at     queue          slots
-----------------------------------------------------------------------------------------
  25753 0.57052 no_admix4- miguel.omull r     03/27/2017 09:57:33 ceab@node112      20

user@cluster-ceab:~/Structure_pruebas$ qstat -u "*"
job-ID  prior   name       user         state submit/ start at     queue          slots
-----------------------------------------------------------------------------------------
  25527 0.50500 QLOGIN     j.palmer     r     03/23/2017 11:42:56 ceab@node106       1
  25605 0.50500 colony.sh  htorrado     r     03/24/2017 11:59:59 ceab@node103       1
  25613 0.53603 GenAlgTE10 vicente.jime r     03/24/2017 14:58:29 ceab@node110      10
  25753 0.57052 no_admix4- miguel.omull r     03/27/2017 09:57:33 ceab@node112      20
  25805 0.57052 test       jmreyes      r     03/27/2017 16:19:37 ceab@node111      20
  25536 0.60500 GenAlgTE64 vicente.jime qw    03/23/2017 12:00:10                   30
```

qlogin

Podemos acceder a un nodo aleatorio con el comando qlogin

```
~$ qlogin
```

Si queremos comprobar que un trabajo esté paralelizando nos sirve como puerta de entrada al nodo donde esté corriendo. Una vez ahí ejecutamos el comando htop.

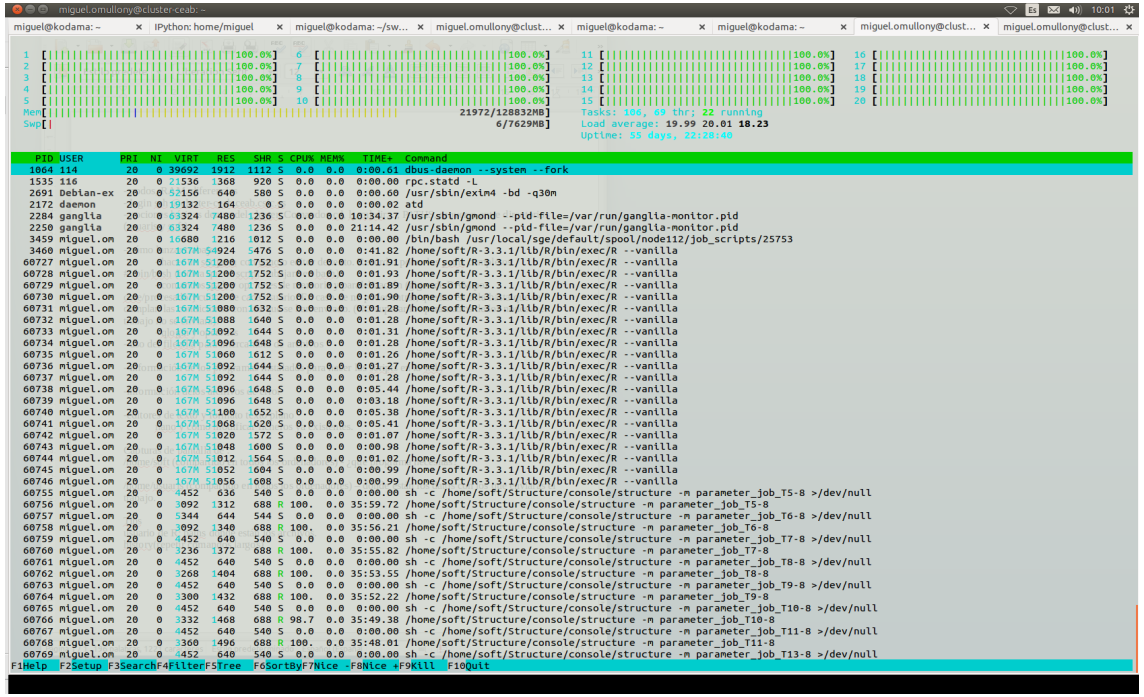
```
~$ ssh nodexxx
user@nodexxx:~$ htop
```

En caso de que no se muestre htop primero escribimos
`~$ export TERM=xterm`

Para salir del nodo y del qlogin

```
nodexxx:~$ exit → ~$ exit → user@cluster-ceab:~$
```

```
miguel.omullony@cluster-ceab:~$ qlogin
[...]
miguel.omullony@node101:~$ ssh node112
[...]
miguel.omullony@node112:~$ htop
```



htop del nodo 112, paralelizando con 20 procesadores

qlogin

También podemos usar el qlogin como un qsub, reservando memoria y procesador.

```
qlogin -l h_vmem=10G -pe make 8
```

Si hacemos un **qstat** (en otra terminal desde cluster-ceab) veremos que tenemos un trabajo QLOGIN en un nodo, en este caso 8 procesadores y 80 gigas de RAM.

Podemos usarlo para hacer pruebas, por ejemplo lanzamos R-3.3.1

```
~$ /home/soft/R-3.3.1/bin/R
```

Y trabajamos con R en línea de comandos

```
user@cluster-ceab:~$ qlogin
local configuration cluster-ceab not defined - using
global configuration
Your job 25892 ("QLOGIN") has been submitted
[...]
user@node108:~$ /home/soft/R-3.3.1/bin/R

R version 3.3.1 (2016-06-21) -- "Bug in Your Hair"
Copyright (C) 2016 The R Foundation for Statistical
Computing
Platform: x86_64-pc-linux-gnu (64-bit)

[...]

Type 'q()' to quit R.

> library(parallel)
```

Si ejecutamos un programa desde cluster-ceab es bastante probable que no funcione. cluster-ceab es una máquina virtual que sirve como entrada al clúster, con muy pocos recursos de RAM y procesador.

Recomendaciones

- El tabulador ahorra tiempo, permite autocompletar palabras o paths.
- Si queremos modificar un comando ya lanzado podemos buscarlo con el comando **history** o con la flecha arriba.
- Consulta la lista de los programas instalados en /home/soft en la *wiki del CBLab* para ver las instrucciones de los scripts por si hace falta indicar el path de alguna librería específica.
- Cuando se lanzan los trabajos se generan en la home diversos archivos con el nombre del script.sh seguidos de una o, e, po, pe (la p si se ha lanzado el trabajo en entorno de paralelización) y el Id job. A veces podemos obtener información de esos archivos.