NGScloud2 v2.13

NGS data analysis using cloud computing

GI Sistemas Naturales e Historia Forestal
(formerly known as GI Genética, Fisiología e Historia Forestal)
Dpto. Sistemas y Recursos Naturales
ETSI Montes, Forestal y del Medio Natural
Universidad Politécnica de Madrid

https://github.com/ggfhf/

Table of contents

Disclaimer 1
Introduction 2
Installation 8
NGScloud2 installation 8
Additional software installation and dependencies 9
Ubuntu Linux 18.04 and 20.04 9
Microsoft Windows 10 10
Mac OS X 13.10 12
First steps 15
Connect to your AWS account 15
Search the Account Id 16
Create an Access Key Id and Secret Access Key 16
Verify email addresses in the Simple Email Service 17
Starting NGScloud2 19
NGScloud2 Docker 20
Configuring your first NGScloud2 environment 21
Consulting submitted processes and troubleshooting 27
A step by step example 29
Create volumes 30
Link volumes in the environment 31
Create cluster with t3.micro instance type in native cluster mode 33
Upload the read files to the cluster 34
Install bioinformatic applications in the cluster 40
Review the quality of reads using FastQC 40
Download the quality analysis results 47
Trim reads using Trimmomatic 50
Terminate the current cluster and create another one with an r5.4xlarge instance type in native cluster mode 55
Assembly reads using Trinity 58
Evaluate the transcriptome using RSEM-EVAL 66
Terminate the current cluster and create another one with r5.xlarge instance type in native cluster mode 73
Transcriptome filtering using transcript-filter 76

Transcriptome clustering using CD-HIT-EST 83
Terminate the current cluster and create another one with c4.xlarge instance type in StarCluster cluster mode 90
Upload the protein database to the cluster 94
Add nodes to the cluster PcanCIC-c4.xlarge 98
Annotate the filtered and clustered transcriptome using transcriptome-blastx 102
Terminate the current cluster and create another one with t3.micro instance type in native cluster mode 109
Download the transcriptome, evaluation and annotation files 112
Terminate the cluster with t3.micro instance type 125
Use of the M4, C4 and R4 instance types in StarCluster mode 127
Application of the workaround 127
Ubuntu Linux and Mac OS X 128
Windows 129
Emergency procedure to terminate a cluster 131
Enlarging a volume 133
How-to… 137
How to display this manual 137
How to recreate the NGScloud2 config file 137
How to create a new environment 137
How to change to another environment 137
How to view characteristics of the instance types 137
How to create a cluster 137
How to terminate a cluster 137
How to list the running clusters 137
How to create a volume 137
How to remove a volume 137
How to list the created volumes 137
How to links volumes in the environment configuration 137
How to add a node in a cluster (only StarCluster cluster mode) 137
How to remove a node in a cluster (only StarCluster cluster mode) 137
How to open a terminal of a cluster 138
How to set up a bioinformatic software in a cluster 138
How to run a *De novo* RNA-seq software in a cluster 138
How to run a Reference-based RNA-seq software in a cluster 138
How to run a RAd-seq software in a cluster 139

How to display datasets of a volume 139
How to display the contents of a dataset 139
How to upload reference files to a cluster 139
How to compress/decompress reference files in a cluster 139
How to upload database files to a cluster 139
How to compress/decompress database files in a cluster 140
How to upload read files to a cluster 140
How to compress/decompress read files in a cluster 140
How to download results files from a cluster 140
How to compress/decompress results files in a cluster 140
How to view submission logs in the local computer 140
How to view result logs in the cluster 140
How to cite 141

Disclaimer

The software package NGScloud2 is available for free download from the GitHub software repository (https://github.com/GGFHF/NGScloud) under GNU General Public License v3.0.

The services of Amazon Web Services, such as Elastic Compute Cloud (EC2), Elastic Block Store (EBS) and Simple Email Service (SES), that uses NGScloud2 entails expenses. You can inform yourself on the Amazon website: htpps://aws.amazon.com. The NGScloud2 development team does not assume any responsibility or liability for the expenses derived from the utilization of the services provided by Amazon Web Services by the NGScloud2 users.

Introduction

NGScloud2, a major version of NGScloud, is a bioinformatic system developed to analyze data of NGS experiments using the cloud computing services of Amazon - Elastic Compute Cloud (EC2)- that permit the access to ad hoc computing infrastructure scaled according to the complexity of each experiment, so its costs and times can be optimized. The application provides a user-friendly front-end to easily operate Amazon's hardware resources and to control workflows of bioinformatic analysis of *de novo* RNA-seq (Figure 1), reference-based RNA-seq (Figure 2), RAD-seq (Figure 3) and functional annotation (Figure 4), specifically oriented to plant species. This latter workflow encapsulates our standalone application Taxonomy-oriented Annotation (https://github.com/GGFHF/TOA/) so it can run in EC2. You can find a complete manual in its website.

NGSCloud2 philosophy is based on the "cluster" concept. A cluster is a set of 1 to n virtual machines with the same instance type. Each instance type has its hardware features: processor type, CPU number, memory amount, etc. (https://aws.amazon.com/ec2/instance-types/).

When a cluster is created, we have to set the cluster mode. There are two cluster modes: native and StarCluster. The StarCluster mode uses StarCluster (http://star.mit.edu/cluster/), an open source cluster-computing toolkit for EC2, which implements clusters of up to 20 virtual machines, enabling faster analysis. The Starcluster software can only use AWS's previous generation instance types, i.e. M3, C3 or R3. In NGScloud2, we provide a patch to enable using instance types M4, C4 and R4 (see the chapter "Use of the M4, C4 and R4 instance types in mode StarCluster). The native mode only manages one virtual machine, either from the previous and current generations, including the last M5, C5 and R5 instance types. A very important advantage of the native cluster mode is the possibility of using instance types named "spot instances" that derive from unused EC2 capacity in the AWS cloud. Using spot instances can lead to significant cost savings (https://aws.amazon.com/ec2/spot/).

When a cluster is created in the StarCluster mode, the first created instance is named "master". Then "subsidiary nodes" can be attached to the "master" node if they are necessary to run some processes in parallel. In this case, each job submitted will be run in the selected node according to the workload.

Data volumes provided by the Amazon Elastic Block Store (EBS) (https://aws.amazon.com/es/ebs/) allow to save and keep data even if there is not any cluster running. NGScloud2 uses five dataset types:

- Applications, located in /ngscloud/apps, where the bionformatic applications are installed.
- Databases, located in /ngscloud/databases, where data from the genomic databases used by the annotation processes are stored.
- Reads, located in /ngscloud/reads, where the raw and pre-processed NGS reads are stored.
- References, located in /ngscloud/references, where genomes, transcriptomes and information about gene structure used by some applications are stored.
- Results, locate in /ngscloud/results, where the output files and logs of the NGScloud2 processes are stored.

In order to hold these dataset types, NGScloud2 offers the possibility of using only a data volume for all dataset types or a volume per each dataset type.

Figure. 1. *De novo* RNA-seq workflow in NGScloud2. The bioinformatics applications that can be used in each step are shown in italics.

Figure. 2. Reference-based RNA-seq workflow in NGScloud2. The bioinformatics applications that can be used in each step are shown in italics.

Figure. 3. RAD-seq workflow in NGScloud2. The bioinformatics applications that can be used in each step are shown in italics.

Figure. 4. Functional annotation workflow in NGScloud2. This workflow encapsules the application Taxonomy-oriented Annotation.

The development of NGScloud2 stems from the needs of specific user-friendly tools for bioinformatic analysis of NGS experiments in small laboratories, or by researchers that lack advanced knowledge in the bioinformatic analysis. NGScloud2 is specially oriented to experiments involving many libraries and massive data generation is expected. NGScloud2 was designed to facilitate analyses since the researcher is guided in the choice of the input files for bioinformatic applications and the parameters to be used, encapsulating the complexity of the command line. In addition, NGScloud2 takes advantage of the resources provided by the Amazon Web Services, so it can also be considered as an alternative to private clusters, to perform the analysis, and to store the read files, results, and associated databases.

Installation

NGScloud2 installation

NGScloud2 was programmed in Python 3, and it runs in any computer with an OS that allows for Python 3: Linux, Microsoft Windows, Mac OS X and other platforms.

NGScloud2 is available from the GitHub software repository of the Forest Genetics and Physiology Research Group (https://github.com/GGFHF/NGScloud/), and it is distributed under GNU General Public Licence Version 3.

To download NGScloud2, click on *Code* and then in *Download ZIP*:

To install NGScloud2 on Linux and Mac OS X, simply decompress the NGScloud2-master.zip into a directory, typing the following command in a terminal window:

```
$ unzip NGScloud2-master.zip
```

Then, the execution permissions of the programs must be set by using this command:

```
$ chmod u+x *.py *.sh
```

For Microsoft Windows, simply unzip NGScloud2-master.zip in the usual way.

Additional software installation and dependencies

Python 3 is necessary for a correct functioning of NGScloud2 was programmed in Python 3: therefore Python 3 needs to be installed in the OS. In addition, Python 2 is also necessary to allow the StarCluster mode. While Python 3 is pre-installed in the last versions of Ubuntu Linux, Python 2 needs to be installed by the user (see below). For Microsoft Windows and Mac OS X, both Python 2 as Python 3 require installation.

For Windows, you can download both Python versions from the official website (https://www.python.org/), or use one of the several distributions that include Python along with other software packages for standard bioinformatic analysis. We recommend installing Anaconda (a version corresponding to Python 3.6 or higher). Anaconda is a free cross-platform for Microsoft Windows, Linux and Mac OS X (https://www.continuum.io/). The installation instructions for Anaconda are available on its web site.

If you are a Mac OS X user and you are not sure about how to install Python, we recommend installing Anaconda as well.

To work properly, NGScloud2 needs the following Python packages:

- Tkinter (https://docs.python.org/3.6/library/tkinter.html), the standard Python interface to the Tk GUI toolkit.
- AWS CLI (https://aws.amazon.com/cli/), the AWS Command Line Interface.
- Boto3 (https://boto3.readthedocs.io/), the AWS SDK (Software Development Kit) for Python.
- Paramiko (http://www.paramiko.org/), an implementation of the SSHv2 protocol in Python.
- Requests (https://requests.kennethreitz.org/en/master/), an HTTP library.
- Plotnine (https://plotnine.readthedocs.io/en/stable/), an implementation of a grammar of graphics in Python based on ggplot2
- StarCluster (http://star.mit.edu/cluster/), an open source cluster-computing toolkit for EC2.

Next, we present how to install this additional software in three example environments: a) an Ubuntu Linux 18.04 (Bionic Beaver) and 20.044 (Focal Fossa) where Python 3 is pre-installed in the OS but not Python 2; b) Windows 10 where Python 2 and Python 3 will be installed using Anaconda. c) Mac OS X 10.13 (High Sierra) where both Python versions will be installed using Anaconda, as well.

Ubuntu Linux 18.04 and 20.04

First, you have to open a terminal window and type the following commands to install Python 2, the Python 3 modules Tk, PIL and PIL.ImageTk, and pip and pip3, if necessary:

```
$ sudo apt install python python-dev

$ wget https://bootstrap.pypa.io/get-pip.py -O get-pip.py && python2 get-pip.py && rm get-pip.py

$ sudo apt install python3-pip python3-tk python3-pil python3-pil.imagetk
```

The additional software may be installed by typing the following commands in the terminal window.

- AWS CLI:

```
$ sudo pip3 install awscli
```

- Boto3:

```
$ sudo pip3 install boto3
```

- Paramiko:

```
$ sudo apt install build-essential libssl-dev libffi-dev python3-dev

$ sudo pip3 install cryptography

$ sudo pip3 install paramiko
```

- Requests:

```
$ sudo pip3 install requests
```

- Plotnine:

```
$ sudo pip3 install plotnine
```

- StarCluster:

```
$ export PATH=~/.local/bin:$PATH

$ pip install starcluster
```

Microsoft Windows 10

Assuming that Anaconda has been installed in Windows with Python 3.6 or higher as main environment, Python 2.7 must be installed as additional environment identified as *py27* running the following command on a Command Prompt started as Administrator:

```
> conda create --name py27 python=2.7 anaconda
```

Tkinter and Requests modules are contained in the Anaconda installation. Then, we can install the remaining Python modules typing in the same Command Prompt started as Administrator:

- AWS CLI:

```
> pip install awscli
```

- Boto3:

```
> conda install boto3
```

- Paramiko:

```
> conda install paramiko
```

- Plotnine:

```
$ conda install --channel conda-forge plotnine
```

- StarCluster:

```
> conda activate py27

> conda install pycrypto

> pip install starcluster

> conda deactivate
```

If *Anaconda3_path* is the directory where you have installed Anaconda3, you must review the window "Edit environment variable" in "System Properties" and verify that the following directories are declared as PATH variables:

- *Anaconda3_path*
- *Anaconda3_path\Library\mingw-w64\bin*
- *Anaconda3_path\Library\usr\bin*
- *Anaconda3_path\Library\bin*
- *Anaconda3_path\Scripts*
- *Anaconda3_path\envs\py27\Scripts*

You can see below an example of this windows.

Mac OS X 13.10

First, we have to install Homebrew and wget commands, if necessary, typing the following commands in the terminal window:

```
$ /usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"

$ brew install wget
```

Now we download the Anaconda software file, e.g. the version 2020.02, typing the command:

```
$ wget https://repo.anaconda.com/archive/Anaconda3-2020.02-MacOSX-x86_64.sh
```

And we give execution permission to this file and run it typing the commands:

```
$ chmod u+x Anaconda3-2020.02-MacOSX-x86_64.sh

$ ./Anaconda3-2020.02-MacOSX-x86_64.sh
```

During the Anaconda installation, we have to read the Anaconda End User License Agreement, accept the license terms and indicate the location where Anaconda will be installed. Later, we must review that the PATH variable incorporates the directory where Python is. So, if Anaconda installation directory is *Anaconda3_path*, we edit *.bash_profile* file in the user's

home directory and check the directory *Anaconda3_path/bin*. is added in the PATH variable through a sentence like this:

```
export PATH=Anaconda3_path/bin:$PATH
```

The appearance of the file *.bash_profile* must be similar to the file shown below:

```
# added by Anaconda3 2.3.0 installer
export PATH="/Users/jmoramarquez/bioinfo/anaconda3/bin:$PATH"

# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/Users/jmoramarquez/bioinfo/anaconda3/bin/conda' 'shell.bash' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/Users/jmoramarquez/bioinfo/anaconda3/etc/profile.d/conda.sh" ]; then
        . "/Users/jmoramarquez/bioinfo/anaconda3/etc/profile.d/conda.sh"
    else
        export PATH="/Users/jmoramarquez/bioinfo/anaconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<
```

Now, Python 2.7 must be installed as additional environment identified as *py27* typing the following command on the terminal window:

```
$ conda create --name py27 python=2.7 anaconda
```

Tkinter and Requests modules are contained in the Anaconda installation. Then, we can install the remaining Python modules typing on the terminal:

- AWS CLI:

```
$ pip install awscli
```

- Boto3:

```
$ conda install boto3
```

- Paramiko:

```
$ conda install paramiko
```

- Plotnine:

```
$ conda install --channel conda-forge plotnine
```

- StarCluster:

```
$ conda activate py27

$ conda install pycrypto
```

```
$ pip install starcluster

$ conda deactivate
```

Then we have to modify the *.bash_profile* file to include in the PATH variable the directory *Anaconda3_path/envs/py27/bin*:

```
export PATH=Anaconda3_path/bin:Anaconda3_path/envs/py27/bin:$PATH
```

The new appearance of the file .bash_profile must be similar to the file shown below:

```
# added by Anaconda3 2.3.0 installer
export PATH="/Users/jmoramarquez/bioinfo/anaconda3/bin:/Users/jmoramarquez/bioinfo/anaconda3/envs/py27/bin:$PATH"

# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/Users/jmoramarquez/bioinfo/anaconda3/bin/conda' 'shell.bash' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/Users/jmoramarquez/bioinfo/anaconda3/etc/profile.d/conda.sh" ]; then
        . "/Users/jmoramarquez/bioinfo/anaconda3/etc/profile.d/conda.sh"
    else
        export PATH="/Users/jmoramarquez/bioinfo/anaconda3/bin:/Users/jmoramarquez/bioinfo/anaconda3/envs/py27/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<
```

First steps

The following steps are mandatory before you can use NGScloud2, after NGScloud2 and the additional software have been installed:

- Connect to your AWS Account
- Search the Account Id
- Create an Access Key Id and Secret Access Key
- Verify e-mail addresses in the Simple Email Service
- Start NGScloud2
- Configuring your first NGScloud2 environment

Connect to your AWS account

First, you must connect to your AWS Account in the web site htpps://aws.amazon.com clicking in *Sign in to the Console*:

Then, complete your e-mail and password personal information in the corresponding text boxes:

If you don't have an AWS Account, you can create one. Currently, Amazon allows the users the access to restricted services for free for one year. Information about how to use the free tier is properly explained in: http://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/billing-free-tier.html.

Search the Account Id

The Account Id is a 12 digits' number located in *My Account* information:

Create an Access Key Id and Secret Access Key

In *Your Security Credentials*, you must click on *Access Keys (Access Key Id and Secret Access Key)* option. Then, new information will be displayed, and you must click on *Create New Access Key)*.

Next, a dialog box will display to confirm that your Access Key Id and Secret Access Key have been created successfully. Once this has been checked, you must download a file containing your personal Access Key Id and Secret Access key by clicking in *Download Key File* button.

Verify email addresses in the Simple Email Service

NGScloud2 automatically sends the users an e-mail to report when a process has ended using the Simple Email Service (SES). When you first sign up for SES, your account is the sandbox access Only 200 e-mails can be sent in a 24-hour period at a maximum rate of one email per second using the sandbox access. In the unlikely case that you run more than 200 NGScloud2 processes per day (i.e. more than 200 e-mail are to be submitted), it is possible to increase your sending quotas. (see SAS instructions at https://docs.aws.amazon.com/ses/latest/DeveloperGuide/manage-sending-quotas-request-increase.html).

SES requires that you verify the email addresses that you want to use. Open the SES console at https://console.aws.amazon.com/ses/ and in the region selector, choose the AWS Region

where want to verify the email address. NGScloud2 uses us-east-1, US East (N. Virginia) by default, but you can modify it.

Click the menu item *Email Addresses* and a new window will appear:

Click the button *Verify a New Email Address* and a dialog box will appear:

Then, type your e-mail address in the corresponding text box and click the button *Verify This Email Address*. You will receive a message in order to verify this email address.

These steps can be repeated to verify more than one e-mail address.

Starting NGScloud2

You must set the directory where NGScloud2.zip was decompressed as the current directory in a terminal window or command prompt. NGScloud2 will run in graphical mode using the graphical user interface (GUI), but it can also be run in console mode on server machines without GUI installed. Here, we explain how to run NGScloud2 in GUI mode. However, the console mode has menus with the same options available in GUI mode.

If you are a Linux or Mac OS X user, you start NGScloud2 in GUI mode typing the following command in a terminal window in the directory where the package of NGScloud2 is downloaded:

```
$ ./NGScloud2.py
```

Alternatively, you can type also:

```
$ ./NGScloud2.py --mode=gui
```

To run NGScloud2 in console mode:

```
$ ./NGScloud2.py --mode=console
```

The file NGScloud2.bat allows to execute NGScloud2.py to the Microsoft Windows users calling the Python interpreter. You may have to modify the value of some environment variables in NGScloud2.bat for its proper work. Then, type the following command to run NGScloud2 in a Command Prompt in the directory where the package of NGScloud2 is downloaded:

```
> NGScloud2
```

You can type too:

```
> NGScloud2 --mode=gui
```

And to run NGScloud2 in console mode, type the command:

```
> NGScloud2 --mode=console
```

NGScloud2 Docker

If you are a Docker user, you can run the NGScloud2 image typing using a Bash script with the following instructions:

```
LOCAL_CONFIG_DIR=/home/$USER/Docker-NGScloud2/config
LOCAL_KEYPAIRS_DIR=/home/$USER/Docker-NGScloud2/keypairs
LOCAL_LOGS_DIR=/home/$USER/Docker-NGScloud2/logs
LOCAL_TEMP_DIR=/home/$USER/Docker-NGScloud2/temp
LOCAL_SWAP_DIR=/home/$USER/Docker-NGScloud2/swap

mkdir --parents $LOCAL_CONFIG_DIR
mkdir --parents $LOCAL_KEYPAIRS_DIR
mkdir --parents $LOCAL_LOGS_DIR
mkdir --parents $LOCAL_TEMP_DIR
mkdir --parents $LOCAL_SWAP_DIR

xhost + local: 2>&1 > /dev/null

docker run \
    --name ngscloud2-container \
    --volume /tmp/.X11-unix:/tmp/.X11-unix \
    --volume "$LOCAL_CONFIG_DIR":/Docker/NGScloud2/config \
    --volume "$LOCAL_KEYPAIRS_DIR":/Docker/NGScloud2/keypairs \
    --volume "$LOCAL_LOGS_DIR":/Docker/NGScloud2/logs \
    --volume "$LOCAL_TEMP_DIR":/Docker/NGScloud2/temp \
    --volume "$LOCAL_SWAP_DIR":/Docker/NGScloud2/NGScloud2-swap \
    --env DISPLAY=unix$DISPLAY \
    --device /dev/snd \
    fernandomoramarquez/ngscloud2
```

The variables LOCAL_CONFIG_DIR, LOCAL_KEYPAIRS_DIR, LOCAL_LOGS_DIR, LOCAL_TEMP_DIR and LOCAL_SWAP_DIR are used to link the TOA container directories with local directories in order to hold the data persistence if you remove the container.

The variable LOCAL_SWAP_DIRECTORY is used to pass files to be processed by NGScloud2, such as read files.

The default values of these four variables can be modified to fit to your preferences.

Configuring your first NGScloud2 environment

Before starting an experiment in NGSCloud2, it is very important to estimate the maximum size each dataset type that we will need, particularly for the reads and results datasets. The read datasets will store the uploaded read files and new read dataset obtained after trimming, if needed. The results datasets will store the outputs of bioinformatic application runs and they may have big size. Also we have to determine if we are going to hold all dataset types in only a data volume or we are going to use a volume per each dataset type.

An "environment" identifies a user, a set of data volume and the AWS zone where processes run and data volumes are stored. When starting NGScloud2 for the first time, you must type the name of the environment (alphanumeric characters only) in the box *Environment*. E. g. **PcanCIC**:

./

In the next window, you must type your *AWS user id*, *access key id* and *secret access key*. A *contact e-mail* address is required too. This e-mail address is used to warn you when a submitted job ends.

Once the first steps are completed, the main window of NGSCloud2 is shown, and it is ready to use.

NGSCloud2 is structured in several menus:

System menu

Just to exit the application.

Cloud control menu

This menu contains all the items related to:

- Setting an environment
- NGScloud2 configuration and security
- Creation of clusters, nodes and volumes, and options to operate with them
- Installation of bioinformatic applications in a cluster
- Open a terminal in a cluster node

De novo RNA-seq menu

Options related to *de novo* RNA-seq experiments:

- Read quality
  - FastQC
- Trimming
  - cutadapt
  - Trimmomatic
- Digital normalization
  - insilico_read_normalization (Trinity package)
- Assembly
  - SOAPdenovo-Trans
  - Trans-ABySS
  - Trinity
- Read alignment
  - Bowtie2
  - GSNAP (GMAP-GSNAP package)
- Transcriptome quality assessment
  - BUSCO
  - QUAST
  - rnaQUAST
  - RSEM-EVAL (DETONATE package)
  - Transrate
- Transcriptome filtering
  - CD-HIT-EST (CD-HIT package)
  - transcript-filter (NGShelper package)
- Quantitation
  - eXpress
  - kallisto
- Annotation
  - transcriptome-blastx (NGShelper package)

Reference-based RNA-seq menu

Options related to reference-based RNA-seq experiments:

- Read quality
  - FastQC
- Trimming
  - cutadapt
  - Trimmomatic
- Assembly
  - Cufflinks-Cuffmerge (Cufflinks package)
  - Genome-guided Trinity (Trinity package)
- Read alignment
  - Bowtie2
  - GSNAP (GMAP-GSNAP package)
  - HISAT2
  - STAR
  - TopHat
- Transcriptome alignment
  - GMAP (GMAP-GSNAP package)
- Quantitation
  - Cuffquant (Cufflinks package)
  - htseq-count (HTSeq package)
- Differential expression
  - Cuffdiff (Cufflinks package)
  - Cuffnorm (Cufflinks package)
- Variant calling
  - Variant calling

RAD-seq menu

Options related to RAD-seq experiments:

- Maintenance ofdata files
  - Restriction site file
  - End file
  - Individual file
  - VCF sample file
  - RADdesigner condition file
- Enzyme analysis
  - Rsitesearch (ddRADseqTools package)
- *In silico* simulations
  - ddRADseq simulation (ddRADseqTools package)
- Read quality
  - FastQC
- RAD design
  - RADdesigner

- Trimming
  - cutadapt
  - Trimmomatic
- Pseudo assembly
  - SOAPdevovo2
  - starcode
- Read alignment
  - Bowtie2
  - GSNAP (GMAP-GSNAP package)
  - HISAT2
- Variant calling
  - Variant calling
- Pipelines
  - ipyrad

Taxonomy-oriented annotataion menu

Options related to the functional annotation:

- Configuration
  - Recreate TOA config file
  - View TOA config file
  - Recreate TOA database

- Genomic databases
  - Basic data
  - Gymno PLAZA 1.0
  - Dicots PLAZA 1.0
  - Monocots PLAZA 1.0
  - NCBI RefSeq Plant
  - NCBI Plant database NT
  - NCBI Nucleotide GenInfo Viridiplantae identifier list
  - NCBI Plant database NR
  - NCBI Protein GenInfo Viridiplantae identifier list
  - NCBI Gene
  - InterPro
  - Gene Ontology
- Annotation pipelines
  - TOA nucleotide pipeline
  - TOA amino acid pipeline
- Statistics
  - Alignment
  - Annotation datasets
  - Species
  - Family
  - Phylum
  - EC

- Gene Ontology
- InterPro
- KEGG
- MapMan
- MetaCyc

Datasets menu

The options included here allow to handle the read, reference, database and results datasets:

- List dataset
- Upload read files from local computers to a cluster.
- Download the results files from a cluster to the local computer.
- Compress and decompress files in a cluster.
- Remove datasets.

Logs menu

This menu allows the access to logs of submissions in the local computer and logs of results in the clusters.

Help menu

It contains the documentation of the application.

Before using any of the options in the menus, "key pairs" need to be created. Key pairs are used to encrypt and decrypt login information. You can create key pairs, by selecting the menu item with this path:

Main menu > Cloud control > Security > Create key pairs

A dialog box will be raised to confirm the action.

A key pair is valid for all zones within a region. Then if you have created a key pair in a zone, and you change to another zone of the same region, you do not have to create the key pair again.

Consulting submitted processes and troubleshooting

The correct operability of the submitted processes is controlled by logs similar to this example:

So, the user can monitor the performance of the process at any time. When a process stops, as a result of the correct ending of the process or by a crash of the program, a mail like this is sent to the contact e-mail address of the user by the system informing of the completion of the process and if it ended OK or with an execution error:

If there is a crash, the user will have to investigate the reasons, initialize clusters with instances of superior performance, if necessary, or revise the input files and the parameters in the config files. Each time a cluster is started is at a monetary cost, so it is advisable to dimension the cluster size (in terms of RAM and CPUs) needed to run each process with success before starting an experiment.

In the next chapter, we present a complete example about how to operate NGScloud2 in order to analyze the results of an RNA-seq experiment, emphasizing the use of the appropriate instance type in each task and the follow-up of processes through their corresponding log.

A step by step example

We have sequencing data corresponding to an RNA-seq Illumina library of an experiment about the process of cicatrization after wounding the xylem of the stem of the Canary Island pine (*Pinus canariensis*). The next table shows the size characteristics of the read files yield by the NGS platform:

Library	File	Read number	Compressed size (in B)	Decompressed size (in B)
Pcan-CIC	Pcan-CIC_1.fastq.gz	21.444.414	1.648.808.931	5.178.672.000
	Pcan-CIC_2.fastq.gz	21.444.414	1.632.988.106	5.178.672.000
	2	**42.888.828**	**3.281.797.037**	**10.357.344.000**

In this example, we are going to review the quality of reads, to trim read ends with bad scores and to assembly the reads yielding a transcriptome.

The steps are the following:

- Create volumes
- Link volumes in the environment configuration
- Create a cluster with the t3.micro instance type in native cluster mode
- Upload the read files to the cluster
- Install the bioinformatic applications in the cluster
- Review the quality of reads using FastQC
- Trim the reads using Trimmomatic
- Terminate the current instance and create another one with a r5.4xlarge instance type in native cluster mode
- Assembly the reads using Trinity
- Evaluate the transcriptome quality using RSEM-EVAL
- Terminate the current cluster and create another one with a r5.xlarge instance type in native cluster mode
- Transcriptome filtering using transcript-filter
- Transcriptome clustering using CD-HIT-EST
- Terminate the current cluster and create another one with a c4.xlarge instance type in StarCluster cluster mode
- Upload the protein database to the cluster
- Add nodes to the cluster with c4.xlarge instance type
- Annotate the filtered and clustered transcriptome using transcriptome-blastx
- Terminate the current cluster and create another one with a t3.micro instance type in native cluster mode
- Download the transcriptome, evaluation and annotation files
- Terminate the cluster with t3.micro instance type

Create volumes

In this example, we are going to use a volume per each dataset type. We need to create the data volumes to have persistent storage of the installed bioinformatic applications, databases, reads, references and results. We have to decide the type and the size of each volume. First, ten GiB can be enough size for the applications volume. In this case, we choose a standard HDD type, given the size and the cost per GiB of each volume type. To create the volume, select the menu item with this path:

Main menu > Cloud control > Volume Operation > Create volume

In the raised window, we type **PcanCIC-apps** in *Volume name* textbox, we select **standard HDD** in *Volume type* combo-box, and we type **10** in *Volume size (in GiB)* textbox; we untick *Terminate volume creator?* checkbox; and we press the *Execute* button:

A volume creator instance will be started to create and format the volume. When the volume is created and formatted, the volume creator will not be terminated (we have unticked *Terminate volume creator?* checkbox), which will allow us to create the other volumes quickly.

The names of reads, reference, database and results volumes are **PcanCIC-reads**, **PcanCIC-references**, **PcanCIC-databases** and **PcanCIC-results**, respectively. The sizes of reads,

reference, database and results volumes are **20** GiB, **5**GiB, 5GiB and **100** GiB, respectively. The volume type is a **standard HDD** for all cases. We repeat the steps done to configure the app volume, to configure the reads and results volumes, making sure that the flag of *Terminate volume creator?* checkbox must be ticked when creating our last volume.

We can review the created volumes selecting the menu item as follows:

Main menu > Cloud control > Volume Operation > List volumes

The pop-up window will show information about the created volumes:

Zone	Volume Name	Volume Id	Size (GiB)	State	Attachments
us-east-1c	PcanCIC-apps	vol-06ca1b476b0410bf1	10	available	0
us-east-1c	PcanCIC-databases	vol-0a91e05f13ba61536	5	available	0
us-east-1c	PcanCIC-reads	vol-00eee59f776a0fd83	20	available	0
us-east-1c	PcanCIC-references	vol-0628624c8d1eeff88	5	available	0
us-east-1c	PcanCIC-results	vol-04392abe34f0e9762	100	available	0

Link volumes in the environment

We must link the created volumes in the environment configuration, so that the volumes are automatically attached at the start of the cluster. Select the menu item with this path:

Main menu > Cloud control > Configuration > Link volumes

In the raised window, we select **multi-volume** in *Data structure* combo-box. Then, *Application volume*, *Database volume*, *Read volume*, *Reference volume* and *Result volume* combo-boxes are enable and the NGScloud volume combo-box, used when the dataset structure is single-volume, is disabled. We select **PcanCIC-apps**, **PcanCIC-databases**, **PcanCIC-reads**, **PcanCIC-references**, and **PcanCIC-results** in *Application volume*, *Database volume*, *Read volume*, *Reference volume* and *Result volume* combo-boxes, respectively. Then we have to press the *Execute* button.

A pop-up window will display the run log:

Create cluster with t3.micro instance type in native cluster mode

Now we create a cluster with a t3.micro instance type, 2 CPUs and 1 GiB of RAM, because the read file upload and the read trimming require small hardware resources. We select the menu item with this path:

Main menu > Cloud control > Cluster operation > Create cluster

In the raised window, we select **native** in the *Cluster mode* combo-box and press the button close to *Instance type* textbox. Then, we double click on the t3.micro instance type in the pop-up window that contains the instance type list. Now, **t3.micro** is in the *Instance type* textbox. The rest of the parameters keep the default value. Finally, we press the *Execute* button:

A pop-up window will display the run log:

When the cluster is started, the infrastructure software installation is submitted.

Upload the read files to the cluster

Each task related to datasets or to the run of a bioinformatic program has:

- First, a window to help to select datasets or specific files. A config file is created according to the user selection and default values of the parameters of the program.
- Then, a window with the parameters of the program are shown along with an explanation of its meaning. Every parameter has a default value that can be changed.
- Finally, building of a bash script to run the program using the config file and the submission of this script to the cluster.

To select specific files, the first window has a text box where a pattern must be entered. The pattern must be a Python regular expression. A regular expression is used to find a string in other string(s). The pattern is formed by a sequence of characters; some of them have a special meaning, e.g. "." means "any character except newline" and "*" means "0 or more repetitions of the preceding element". You can learn about Python regular expressions at: https://docs.python.org/3.6/library/re.html

Perhaps, these examples are useful for your selection:

Pattern	Selection
.*	all the files
transcriptome.fasta	the file whose name is "transcriptome.fasta"
.*fastq	all files whose name ended in "fastq"
.*fastq.gz	all files whose name ended in "fastq".gz"
.*Pcan.*	all files whose name contains the characters "Pcan"
.*PCan.*fastq	all files whose name contains the characters "Pcan" and ends in "fastq"

To create a config file to upload the read files to a cluster, select the menu item with this path:

Main menu > Dataset > Read dataset file transfer > Recreate config file

In the raised window, we type **PcanCIC** in the *Experiment id* textbox, the local directory where the files are in the *Local directory* textbox (or we select it using the next button), and we type **.*** as the pattern to select the files in the *File pattern* textbox. Then we press the *Execute* button:

In the pop-up window, we can edit the config file created, and remove files or add new files if the file pattern has not selected the appropriate ones:

This window is a text editor, and data can be easily modified. When we save the configuration file, the modifications are validated. If there are errors, a list of them is displayed.

In this example, we can notice that the config file has three sections: *identification*, with the experiment identification; *file-1*, with the local path of the first read file; and *file-2* with the second one.

It is convenient to perform the file transfer steps when an Internet connection with a large bandwidth is available due to the large size of many of the files necessary to perform full RNA-seq analysis.

To upload the read files to the cluster, we select the menu item with this path:

Main menu > Dataset > Read dataset file transfer > Upload dataset to a cluster

In the raised window, we select **PcanCIC-t3.micro** in *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the upload log:

Now, we are going to review the uploaded files. We select the menu item with this path:

Main menu > Dataset > List dataset

In the raised window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box and **reads** in the *Dataset type* combo-box. Then we press the *Execute* button:

We can check the experiments whose read files have been uploaded in the pop-up window. We click on the **PcanCIC** row:

Now, a pop-up window with the read datasets of the experiment PcanCIC is shown:

So far, we have only one: the dataset corresponding to the uploaded-reads. We click on it and another pop-up window appears with the content of the uploaded-reads dataset. In this case, the two files are shown.

If we click on a file row, e.g. the **Pcan-CIC_1.fastq.gz** one, the characteristics of this file are listed:

Install bioinformatic applications in the cluster

Bioconda is necessary to install the bionformatic applications. To install Bioconda in the application volume in a cluster, we select the menu item with this path:

Main menu > Bioinfo software install > Miniconda3 (Python & Bioconda environment)

And, in the next windows, we type the cluster name.

To install FastQC in the application volume in a cluster, we select the menu item with this path:

Main menu > Bioinfo software install > FastQC

And in the next windows, type the cluster name. Also, we install Trimmomatic, Trinity, RSEM, CD-HIT and NGShelper, as we did for the FastQC installation.

Review the quality of reads using FastQC

Now we are going to review the quality of reads using FastQC. First, we create the configuration file selecting the menu item wit this path:

Main menu > De novo RNA-seq > Read quality > FastQC > Recreate config file

In the raised window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box, **PcanCIC** in the *Experiment id* combo-box, **uploaded reads** in the *Read dataset* combo-box, and we type **.*fastq.gz** as the pattern to select the files in the *File pattern* textbox. Then we press the *Execute* button:

In the pop-up window, we can inspect the config file. In this example, there are four sections: *identification*, with the experiment and the read dataset identifications; *FastQC parameters*, with the thread number parameter of FastQC (we modify its value to **1**); *file-1*, with the local path of the first read file; and *file-2* with the path of the second one:

To run the quality process in the cluster, we select the menu item with this path:

Main menu > De novo RNA-seq > Read quality > FastQC > Run read quality process

In the raised window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the submission log:

We can view the process log during and after the run. To do so, we select the menu item with this path:

Main menu > Logs > View result logs in the cluster

In the raised window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box and **PcanCIC** in the *Experiment id* combo-box; and then we press the *Execute* button:

Now, a pop-up window with the results datasets of each bioinformatic program run in the experiment PcanCIC is displayed:

So far, we have only performed a single run: the dataset **fastqc-170925-172321** corresponding to the last (and unique) FastQC run. Clicking on it, another pop-up window appears with its corresponding log.

In the toolbar, there is a button to refresh the run status. Clicking it, the log will be updated.

All the process logs have:

- A header with the node where the script runs and the time when it started.
- Information about the elapsed time, the CPU usage and the maximum memory is displayed for each run of the bioinformatic program.
- At the bottom, a summary with the status (OK, if all the programs have ended without errors; WRONG, otherwise), the end time, and the duration of the script run.

In order to access a list with the output files generated by FastQC, we select the menu item with this path:

Main menu > Dataset > List dataset

In the raised window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box and **results** in the *Dataset type* combo-box. Then we press the *Execute* button:

A pop-up window will display the list of experiments. We click on the **PcanCIC** row:

Next, another pop-up window with the results datasets of the experiment PcanCIC is shown:

So far, we only have one: the dataset corresponding to the FastQC analysis recently completed. We click on it and another pop-up window appears with the content of the files corresponding to this analysis.

Download the quality analysis results

Next, we are going to review the analysis files generated by FastQC. First, we have to download the ".html" files with the results to a local computer. To do so, we first create the configuration file by selecting the menu item with the following path:

Main menu > Datasets > Result dataset file transfer > Recreate config file

In the raised window, we select **PcanCIC-t3**

.micro in the *Cluster name* combo-box, **PcanCIC** in the *Experiment id* combo-box, **uncompressed** in the *Dataset status* combo-box (because the dataset has not been previously compressed), **fastqc-170925-172321** in the *Result dataset* combo-box, and we type **.*html** as the pattern to select the files in the *File pattern* textbox and the local directory where the files will be downloaded in the *Local directory* textbox (or we select it using the button close to the textbox). Then we press the *Execute* button:

In the pop-up window, we can inspect the config file. It has three sections: *identification*, with the experiment and the results dataset identifications, the compression status of the dataset, and the local directory; *file-1*, with the name of the first result file; and *file-2* with the name of the second one:

To download the result files from the cluster, we select the menu item with this path:

Main menu > Dataset > Result dataset file transfer > Download dataset from a cluster

In the raised window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the log corresponding to the download:

Trim reads using Trimmomatic

Once we have reviewed the two result files, we have decided to cut 12 nucleotides from the start of reads. In this point, we are going to use Trimmomatic to do this step.

First, we create the configuration file by selecting the menu item with this path:

Main menu > De novo RNA-seq > Trimming > Trimmomatic > Recreate config file

In the raised window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box, **PcanCIC** in the *Experiment id* combo-box, **uploaded reads** in the *Read dataset* combo-box, we type **.*fastq.gz** as the pattern to select the files in *File pattern* textbox and we select **Paired-end** in the Read type combo-box; finally, we type **1.fastq.gz** in the *File #1 specific chars* textbox and **2.fastq.gz** in the *File #2 specific chars* textbox. These last two strings are used to distinguish the file of each strand among the selected files by the pattern corresponding to the experiment libraries. In this example, there is only one library.

Then we press the *Execute* button:

In the pop-up window, we visualize the config file. In this example, it has six sections: *identification*, with the experiment and the read dataset identifications; *Trimmomatic parameters*, with the thread number (we modify its value to **1**) and phred quality score; *Trimming step values*, with the step list that Trimmomatic can perform (we modify the headcrop value to **12**); *Trimming step order* with the order in which Trimmomatic must carry out every step indicated in the previous section; *library* with the library type; *library-1* with the two read files for the first library (in this example, we only have one library):

We run the trimming process in the cluster by selecting the menu item with this path:

Main menu > De novo RNA-seq > Trimming > Trimmomatic > Run trimming process

In the raised window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the submission log:

We can view the process log during and after its run. To do so, we select the menu item with this path:

Main menu > Logs > View result logs in the cluster

In the raised window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box and **PcanCIC** in the *Experiment id* combo-box; and then we press the *Execute* button:

A pop-up window will display the results datasets for each run of the bioinformatic programs that correspond to the experiment PcanCIC:

At this moment, there are two results datasets: **fastqc-170925-172321**, corresponding to the previous run of FastQC; and **trimmo-170927-202434**, corresponding to the run of Trimmomatic. We click on this last dataset and another pop-up window appears with its corresponding log:

Terminate the current cluster and create another one with an r5.4xlarge instance type in native cluster mode

After read trimming, we are going to assembly a preliminary transcriptome using Trinity. Trinity's hardware requirements are very high in terms of CPUs and GiBs of RAM memory. We must terminate the current cluster and create another one fulfilling these requirements. We choose a r5.4xlarge instance type whose instances have 16 CPUs and 122 GiBs of RAM memory, in order to be able to analyze the large read files of our experiment with Trinity.

To terminate the current cluster, we select the menu item with this path:

Main menu > Cloud control > Cluster operation > Terminate cluster

In the raised window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the run log:

To create the cluster with r5.4xlarge instance type, we select the menu item with this path:

Main menu > Cloud control > Cluster operation > Create cluster

In the raised window, we select **native** in the *Cluster mode* combo-box and press the button close to *Instance type* textbox. Then, we double click on the r5.4xlarge instance type in the pop-up window that contains the instance type list. Now, **r5.4xlarge** is in the *Instance type* textbox. The rest of the parameters keep the default value. Finally, we press the *Execute* button:

A pop-up window will display the run log:

When the cluster is started, the infrastructure software installation is submitted.

Assembly reads using Trinity

First, we create the config file by selecting the menu item with this path:

Main menu > De novo RNA-seq > De novo assembly > Trinity > Recreate config file

In the raised window, we select **PcanCIC-r5.4xlarge** in the *Cluster name* combo-box, **PcanCIC** in the *Experiment id* combo-box, **Trimmomatic (170927 202434)** in the *Read dataset* combo-box, we type **.*fastq.gz** as the pattern to select the files in *File pattern* textbox, and we select **Paired-end** in the *Read type* combo-box; finally, we type **1.fastq.gz** in the *File #1 specific chars* textbox and **2.fastq.gz** in the *File #2 specific chars* textbox. Then we press the *Execute* button:

In the pop-up window, we can examine the config file. In this example, it has four sections: *identification*, with the experiment and the read dataset identifications; *Trinity parameters*, with several parameters used by Trinity in which we can modify the value of CPUs number to **16**, and the value of suggested maximum memory to **100**; *library* with the format and library type; *library-1* with the two read files for the first library (in this example, we only have one library):

We run the assembly process in the cluster by selecting the menu item with this path:

Main menu > De novo RNA-seq > De novo assembly > Trinity > Run assembly process

In the raised window, we select **PcanCIC-r5.4xlarge** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up with the submission log is displayed:

We can view the process log during and after the run. To do so, we select the menu item with this path:

Main menu > Logs > View result logs in the cluster

In the raised window, we select **PcanCIC-r5.4xlarge** in the *Cluster name* combo-box and **PcanCIC** in the *Experiment id* combo-box; and then we press the *Execute* button:

A pop-up window with the results datasets for each run of the bioinformatic programs that correspond to the experiment PcanCIC is displayed:

Experiment id	Result dataset	Bioinfo app / Utility	Date	Time
PcanCIC	fastqc-170925-172321	FastQC	2017-09-25	17:23:21
PcanCIC	trimmo-170927-202434	Trimmomatic	2017-09-27	20:24:34
PcanCIC	trinity-170929-104827	Trinity	2017-09-29	10:48:27

Three results datasets are shown. We click on the **trinity-170929-104827** row, which is the dataset generated by the Trinity run, and another pop-up window appears with its corresponding log:

Now, we are going to list the assembly generated by Trinity. We select the menu item with this path:

Main menu > Dataset > List dataset

In the raised window, we select **PcanCIC-r5.4xlarge** in the *Cluster name* combo-box and **results** in the *Dataset type* combo-box. Then we press the *Execute* button:

In the pop-up window, we click on the **PcanCIC** row:

Next, a pop-up window with the results datasets of the experiment PcanCIC is displayed:

We click on the **trinity-170929-104827** row, and another pop-up window appears with the content of the files corresponding to this assembly.

The file **Trinity.fasta** is the one corresponding to the transcriptome. To observe its characteristics, we click on it:

Evaluate the transcriptome using RSEM-EVAL

Next, we are going to evaluate the quality of the transcriptome generated by Trinity with RSEM-EVAL, which is included in the DETONATE package. To do so, we first create the config file by selecting the menu item with the following path:

Main menu > De novo RNA-seq > Assembly quality and transcript quantification > RSEM-EVAL (DETONATE package) > Recreate config file

In the raised window, we select **PcanCIC-r5.4xlarge** in the *Cluster name* combo-box, **PcanCIC** in the *Experiment id* combo-box, **Trimmomatic (170927 202434)** in the *Read dataset* combo-box, we type **.*fastq.gz** as the pattern to select the files in *File pattern* textbox, and we select **Paired-end** in the *Read type* combo-box; we type **1.fastq.gz** in the *File #1 specific chars* textbox and **2.fastq.gz** in the *File #2 specific chars* textbox; finally, we selected **Trinity (170927 104827)** in the *Assembly dataset* combo-box. The *Assembly type* combo-box only is activated when the assembly dataset was generated by SOAPdenovo-Trans; in this case, the combo-box has two items: CONTIGS and SCAFFOLDS. Then we press the *Execute* button:

In the pop-up window, we can examine the config file. In this example, it has four sections: *identification*, with the experiment, read and assembly dataset identifications; *RSEM-EVAL parameters*, with several parameters used by RSEM-EVAL, where we can modify the value of threads number to **16**; *library* with the format and library type; and *library-1* with the two read files for the first library (in this example, we only have one library):

We run the assembly assessment process in the cluster by selecting the menu item with this path:

Main menu > De novo RNA-seq > Assembly quality and transcript quantification > RSEM-EVAL (DETONATE package) > Run assembly assessment process file

In the raised window, we select **PcanCIC-r5.4xlarge** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up with the submission log is displayed:

We can view the process log during and after its run. To do so, we select the menu item with this path:

Main menu > Logs > View result logs in the cluster

In the raised window, we select **PcanCIC-r5.4xlarge** in the *Cluster name* combo-box and **PcanCIC** in the *Experiment id* combo-box; and then we press the *Execute* button:

A pop-up window with the results datasets for each run of the bioinformatic programs that correspond to the experiment PcanCIC is displayed:

Experiment id	Result dataset	Bioinfo app / Utility	Date	Time
PcanCIC	fastqc-170925-172321	FastQC	2017-09-25	17:23:21
PcanCIC	rsemeval-170930-101146	RSEM-EVAL	2017-09-30	10:11:46
PcanCIC	trimmo-170927-202434	Trimmomatic	2017-09-27	20:24:34
PcanCIC	trinity-170929-104827	Trinity	2017-09-29	10:48:27

Four results datasets are shown. We click on the **rsemeval-170930-101146** row, the dataset generated by RSEM-EVAL run, and another pop-up window is shown with its corresponding log:

Now, we are going to view the assessment files generated by RSEM-EVAL. We select the menu item with this path:

Main menu > Dataset > List dataset

In the raised window, we select **PcanCIC-r5.4xlarge** in the *Cluster name* combo-box and **results** in the *Dataset type* combo-box. Then we press the *Execute* button:

Now, we click on the **PcanCIC** row in the pop-up window:

Next, a new pop-up window with the results datasets of the experiment PcanCIC is shown:

We click on the **rsemeval-170930-101146** row, and another pop-up window appears with the content of the files corresponding to this assembly.

The files whose name end in ".results" have the information about the assembly assessment.

Terminate the current cluster and create another one with r5.xlarge instance type in native cluster mode

Now we are going to terminate the PcanCIC-r5.4xlarge and to create a cluster with an r5.xlarge instance type, 4 CPUs and 30.5 GiB of RAM, because it is not necessary to use an instance with many CPUs and large RAM memory in order to .do the task of filtering.

First, we select the menu item with this path:

Main menu > Cloud control > Cluster operation > Terminate cluster

In the raised window, we select **PcanCIC-r5.4xlarge** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the run log:

Now we create a cluster with a r5.xlarge instance type selecting the menu item with this path:

Main menu > Cloud control > Cluster operation > Create cluster

In the raised window, we select **native** in the *Cluster mode* combo-box and press the button close to *Instance type* textbox. Then, we double click on the r5.xlarge instance type in the pop-up window that contains the instance type list. Now, **r5.xlarge** is in the *Instance type* textbox. The rest of the parameters keep the default value. Finally, we press the *Execute* button:

A pop-up window will display the run log:

When the cluster is started, the infrastructure software installation is submitted.

Transcriptome filtering using transcript-filter

Transcript-filter uses a result of RSEM-EVAL to filter transcripts by length (max or min), or by FPKM or TPM. To do this step, we create the config file by selecting the menu item with this path:

Main menu > De novo RNA-seq > Transcriptome filtering > transcript.filter (NGShelper package) > Recreate config file

In the raised window, we select **PcanCIC-r5.xlarge** in the *Cluster name* combo-box, **PcanCIC** in the *Experiment id* combo-box, and **RSEM-EVAL (170930 101146)** in the *RSEM-EVAL dataset* combo-box. Then we press the *Execute* button:

In the pop-up window, we can examine the config file. There are two sections: *identification*, with the experiment and the RSEM-EVAL dataset identifications; and *transcript-filter parameters*, with the minimum and maximum lengths of transcripts, and the minimum FPKM and TPM values selected by the user:

We run the assembly process in the cluster by selecting the menu item with this path:

Main menu > De novo RNA-seq > Transcriptome filtering > transcript.filter (NGShelper package) > Run transcriptome filtering process

In the raised window, we select **PcanCIC-r.xlarge** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the submission log:

We can view the process log during and after the run. To do so, we select the menu item with this path:

Main menu > Logs > View result logs in the cluster

In the raised window, we select **PcanCIC-r5.xlarge** in the *Cluster name* combo-box and **PcanCIC** in the *Experiment id* combo-box; and then we press the *Execute* button:

A pop-up window will display the results datasets for each run of the bioinformatic programs that correspond to the experiment PcanCIC:

Experiment id	Result dataset	Bioinfo app / Utility	Date	Time
PcanCIC	fastqc-170925-172321	FastQC	2017-09-25	17:23:21
PcanCIC	rsemeval-170930-101146	RSEM-EVAL	2017-09-30	10:11:46
PcanCIC	transfil-171011-123558	transcript-filter	2017-10-11	12:35:58
PcanCIC	trimmo-170927-202434	Trimmomatic	2017-09-27	20:24:34
PcanCIC	trinity-170929-104827	Trinity	2017-09-29	10:48:27

Five results datasets are shown. We click on the **transfil-171011-123558** row, which is the dataset generated by the Trinity run, and another pop-up window is presented with its corresponding log:

Now, we are going to list the assembly generated by transcript-filter. We select the menu item with this path:

Main menu > Dataset > List dataset

In the raised window, we select **PcanCIC-r5.xlarge** in the *Cluster name* combo-box and **results** in the *Dataset type* combo-box. Then we press the *Execute* button:

Now, we click on the **PcanCIC** row in the pop-up window:

Next, a pop-up window with the results datasets of the experiment PcanCIC is displayed:

We click on the **transfil-171011-123558** row, and another pop-up window appears with the content of the files corresponding to this assembly.

Type	Name
file	filtered-transcriptome.fasta
file	log.txt
file	transfil-process.sh
file	transfil-starter.sh

The file **filtered-transcriptome.fasta** is the one corresponding to the transcriptome generated by transcript.filter. To inspect its characteristics, we click on it:

Data	Value
directory	/results/PcanCIC/transfil-171011-123558
name	filtered-transcriptome.fasta
size	65181924
permissions	rw-r--r--
modification date	2017-09-11
modification time	10:36
owner group	root
owner name	root

Transcriptome clustering using CD-HIT-EST

First, we create the config file by selecting the menu item with this path:

Main menu > De novo RNA-seq > Transcriptome filtering > CD-HIT-EST (CD-HIT package) > Recreate config file

In the raised window, we select **PcanCIC-r5.xlarge** in the *Cluster name* combo-box, **PcanCIC** in the *Experiment id* combo-box, and **transcript-filter (171011 123538)** in the *Assembly dataset* combo-box. The *Assembly type* combo-box is only activated when the assembly dataset was generated by SOAPdenovo-Trans; in this case, the combo-box has two items: CONTIGS and SCAFFOLDS. Then we press the *Execute* button:

In the pop-up window, we can examine the config file. In this example, it has two sections: *identification*, with the experiment and the assembly dataset identifications; *CD-HIT-EST parameters*, with several parameters used by CD-HIT-EST, where we can modify the value of threads number to **0** (this value indicates that all CPUs sill be used), the value of memory_limit to **0** (this value indicates unlimited value), and the value sequence identity threshold to **0.8** (or any desired value above 0.8):

We run the assembly process in the cluster by selecting the menu item with this path:

Main menu > De novo RNA-seq > Transcriptome filtering > CD-HIT-EST (CD-HIT package) > Run transcriptome filtering process

In the raised window, we select **PcanCIC-r5.xlarge** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the submission log:

We can view the process log during and after the run. To do so, we select the menu item with this path:

Main menu > Logs > View result logs in the cluster

In the raised window, we select **PcanCIC-r5.xlarge** in the *Cluster name* combo-box and **PcanCIC** in the *Experiment id* combo-box; and then we press the *Execute* button:

A pop-up window will display the results datasets for each run of the bioinformatic programs that correspond to the experiment PcanCIC:

Experiment id	Result dataset	Bioinfo app / Utility	Date	Time
PcanCIC	cdhitest-171013-115617	CD-HIT-EST	2017-10-13	11:56:17
PcanCIC	fastqc-170925-172321	FastQC	2017-09-25	17:23:21
PcanCIC	rsemeval-170930-101146	RSEM-EVAL	2017-09-30	10:11:46
PcanCIC	transfil-171011-123558	transcript-filter	2017-10-11	12:35:58
PcanCIC	trimmo-170927-202434	Trimmomatic	2017-09-27	20:24:34
PcanCIC	trinity-170929-104827	Trinity	2017-09-29	10:48:27

Six results datasets are shown. We click on the **cdhitest-171013-115617** row, which is the dataset generated by the CD-HIT-EST run, and another pop-up window will display its corresponding log:

Now, we are going to list the assembly generated by CD-HIT-EST. We select the menu item with this path:

Main menu > Dataset > List dataset

In the raised window, we select **PcanCIC-r5.xlarge** in the *Cluster name* combo-box and **results** in the *Dataset type* combo-box. Then we press the *Execute* button:

Now, we click on the **PcanCIC** row in the pop-up window:

Next, a pop-up window will display the results datasets of the experiment PcanCIC:

We click on the **cdhitest-171013-115617** row, and another pop-up window appears with the content of the files corresponding to this assembly.

The file **Trinity.fasta** is the one corresponding to the transcriptome. To observe its characteristics, we click on it:

Terminate the current cluster and create another one with c4.xlarge instance type in StarCluster cluster mode

We have to read and follow instructions in the chapter "Use of the M4, C4 and R4 instance types in StarCluster mode" before continuing.

Now we are going to terminate the PcanCIC-r5.xlarge and to create a cluster with a c4.xlarge instance type with the same CPU number but with less memory amount because it is not necessary in order to annotate the transcriptome. We are going create the instance in StarCluster cluster mode because we will use 5 instances working together in the annotation process.

First we select the menu item with this path:

Main menu > Cloud control > Cluster operation > Terminate cluster

In the raised window, we select **PcanCIC-r5.xlarge** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the run log:

Now we create a cluster with a c4.xlarge instance type selecting the menu item with this path:

Main menu > Cloud control > Cluster operation > Create cluster

In the raised window, we select **StarCluster** in the *Cluster mode* combo-box and press the button close to *Instance type* textbox. Then, we double click on the c4.xlarge instance type in the pop-up window that contains the instance type list. Now, **c4.xlarge** is in the *Instance type* textbox. Finally, we press the *Execute* button:

A pop-up window will display the run log:

When the cluster is started, the infrastructure software installation is submitted.

Upload the protein database to the cluster

Previously we had downloaded the FASTA protein files from TAIR (The Arabidopsis Information Resource) whose URL is
https://www.arabidopsis.org/download_files/Proteins/Araport11_protein_lists/Araport11_genes.201606.pep.fasta.gz

to the local computer and we had built the database with the program makeblastdb. The name of database file passed to makeblastdb is **Araport11_genes**.

To create a config file to upload the database files to a cluster, select the menu item with this path:

Main menu > Dataset > Database file transfer > Recreate config file

In the displayed window, we type the local directory where the database is in the *Local directory* textbox (or we select it using the next button), **.*** as the pattern to select the files in the *File pattern* textbox, and **TAIR-Araport11_genes** in the *Database* textbox. Then we press the *Execute* button:

In the pop-up window, we can edit the config file created. In this example, we can notice that the configure file has a section *identification*, with the database identification and the local directory where the database is; and several section *file-i*, with the name of each file of the local directory:

To upload the database files to the cluster, we select the menu item with this path:

Main menu > Dataset > Database file transfer > Upload dataset to a cluster

In the raised window, we select **PcanCIC-c4.xlarge** in *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the upload log:

Now, we are going to review the uploaded files. We select the menu item with this path:

Main menu > Dataset > List dataset

In the displayed window, we select **PcanCIC-c4.xlarge** in the *Cluster name* combo-box and **databases** in the *Dataset type* combo-box. Then we press the *Execute* button:

We can check the experiments whose read files have been uploaded in the pop-up window. We click on **TAIR-Araport11_genes** row:

So far, we only have one: the dataset corresponding to the uploaded database. We click on it and another pop-up window will display the content of the uploaded database:

Add nodes to the cluster PcanCIC-c4.xlarge

We are going to use transcriptome-blastx to annotate the transcriptome. This program supports parallelization, so it can use several nodes to increase the run speed. In this example, we are going to add 4 nodes to the cluster **PcanCIC-c4.xlarge**. So, we will have 5 nodes running, the master and the 4 subsidiary nodes, in such a way one node distributing the work to the other 4.

To add a node to a cluster, we select the menu item with this path:

Main menu > Cloud control > Node operation > Add node in a cluster

In the raised window, we select **PcanCIC-c4.xlarge** in the *Cluster name* combo-box and **node01** in the *Dataset type* combo-box. Then we press the *Execute* button:

A pop-up window will display the run log:

We repeat these actions to add the nodes **node02**, **node03** and **node04**.

Now we are going to inspect the cluster composition selecting the menu item with this path:

Main menu > Cloud control > Cluster operation > Show cluster composing

In the raised window, we select **PcanCIC-c4.xlarge** in the *Cluster name* combo-box and press the *Execute* button:

A pop-up window will display the cluster composition and the characteristics of the nodes:

Annotate the filtered and clustered transcriptome using transcriptome-blastx

First, we create the config file by selecting the menu item with this path:

Main menu > De novo RNA-seq > Annotation > transcriptome-blastx (NGShelper package) > Recreate config file

In the raised window, we select **PcanCIC-c4.xlarge** in the *Cluster name* combo-box, **TAIR-Araport11_genes** in the *Database dataset* combo-box, **Araport11_genes** in the *Database file* combo-box, **PcanCIC** in the *Experiment id* combo-box, and **CD-HIT-EST (171013 115617)** in the *Assembly dataset* combo-box. The *Assembly type* combo-box is only activated when the assembly dataset was generated by SOAPdenovo-Trans; in this case, the combo-box has two items: CONTIGS and SCAFFOLDS. Then we press the *Execute* button:

In the pop-up window, we can examine the config file. There are two sections: *identification*, with the database, experiment and assembly identifications; and *transcriptome-blastx parameters*, with the parameters used by transcriptome-blastx. We modify the node number to **4** and the threads number by node to **4** (every node has 4 CPUs):

We run the annotation process in the cluster by selecting the menu item with this path:

Main menu > De novo RNA-seq > Annotation > transcriptome-blastx (NGShelper package) > Run annotation process

In the raised window, we select **PcanCIC-c4.xlarge** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the submission log:

We can view the process log during and after the run. To do so, we select the menu item with this path:

Main menu > Logs > View result logs in the cluster

In the raised window, we select **PcanCIC-c4.xlarge** in the *Cluster name* combo-box and **PcanCIC** in the *Experiment id* combo-box; and then we press the *Execute* button:

A pop-up window will display the results datasets for each run of the bioinformatic programs that correspond to the experiment PcanCIC:

NGScloud - Experiment runs in /results/PcanCIC - Table

Experiment id	Result dataset	Bioinfo app / Utility	Date	Time
PcanCIC	cdhitest-171013-115617	CD-HIT-EST	2017-10-13	11:56:17
PcanCIC	fastqc-170925-172321	FastQC	2017-09-25	17:23:21
PcanCIC	rsemeval-170930-101146	RSEM-EVAL	2017-09-30	10:11:46
PcanCIC	transbastx-171114-133352	transcriptome-blastx	2017-11-14	13:33:52
PcanCIC	transfil-171011-123558	transcript-filter	2017-10-11	12:35:58
PcanCIC	trimmo-170927-202434	Trimmomatic	2017-09-27	20:24:34
PcanCIC	trinity-170929-104827	Trinity	2017-09-29	10:48:27

Seven results datasets are shown. We click on the **transbastx-171114-133353** row, which is the dataset generated by the transcriptome-blastx run, and another pop-up window is presented with its corresponding log:

NGScloud - View - /results/PcanCIC/transbastx-171114-133352/log.txt

```
#######################################
Script started in node master of cluster PcanCIC-c3.xlarge at 2017-11-14 12:33:57 UTC.
#######################################
Running the transcriptome blastx process ...
Creating the node transcript files ...
There are 68297 transcripts in the transcriptome file.
The transcripts files are created.
Creating the control files ...
The control files are created.
Building the blastx process and watcher scripts ...
The blastx process and watcher scripts are built.
Submiting the blastx process scripts ...
Your job 2 ("blastx-00-process.sh") has been submitted
Your job 3 ("blastx-01-process.sh") has been submitted
Your job 4 ("blastx-02-process.sh") has been submitted
Your job 5 ("blastx-03-process.sh") has been submitted
The blastx process scripts are submitted.
Verifing blastx processes ...
2017-11-14 12:34:07 UTC ... STARTING: 04 - RUNNING: 00 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:35:07 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:36:07 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:37:07 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:38:07 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:39:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:40:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:41:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:42:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:43:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:44:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:45:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:46:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:47:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:48:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:49:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:50:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:51:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:52:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:53:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:54:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:55:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:56:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:57:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
2017-11-14 12:58:08 UTC ... STARTING: 00 - RUNNING: 04 - OK: 00 - WRONG: 00 - OTHERS: 00
```

Now, we are going to list the annotation generated by transcriptome-blastx. We select the menu item with this path:

Main menu > Dataset > List dataset

In the displayed window, we select **PcanCIC-c4.xlarge** in the *Cluster name* combo-box and **results** in the *Dataset type* combo-box. Then we press the *Execute* button:

Now, we click on the **PcanCIC** row in the pop-up window:

Next, a pop-up window will display the results datasets of the experiment PcanCIC:

NGScloud - Directory /results/PcanCIC - Table

Type	Name
directory	cdhitest-171013-115617
directory	fastqc-170925-172321
directory	rsemeval-170930-101146
directory	transbastx-171114-133352
directory	transfil-171011-123558
directory	trimmo-170927-202434
directory	trinity-170929-104827

We click on the **transbastx-171114-133353** row, and another pop-up window appears with the content of the files corresponding to this assembly.

NGScloud - Directory /results/PcanCIC/transbastx-171114-133352 - Table

Type	Name
file	annotation.xml
file	blastx-00-annotation.xml
file	blastx-00-control.txt
file	blastx-00-process.sh
file	blastx-00-process.sh.e2
file	blastx-00-process.sh.o2
file	blastx-00-transcripts.fasta
file	blastx-00-watcher.sh
file	blastx-01-annotation.xml
file	blastx-01-control.txt
file	blastx-01-process.sh
file	blastx-01-process.sh.e3
file	blastx-01-process.sh.o3
file	blastx-01-transcripts.fasta
file	blastx-01-watcher.sh
file	blastx-02-annotation.xml
file	blastx-02-control.txt

The file **annotation.xml** is the one corresponding to the complete annotation, after concatenating the annotation files of all nodes. To observe its characteristics, we click on it:

Terminate the current cluster and create another one with t3.micro instance type in native cluster mode

Now we are going to terminate the PcanCIC-c4.xlarge and to create a cluster with a t3.micro template, because it is not necessary to use an instance with many CPUs and large RAM memory in order to download them to our local computer.

First, we select the menu item with this path:

Main menu > Cloud control > Cluster operation > Terminate cluster

In the raised window, we select **PcanCIC-c4.xlarge** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the run log:

Now we create a cluster with a t3.micro instance type selecting the menu item with this path:

Main menu > Cloud control > Cluster operation > Create cluster

In the raised window, we select native in the Cluster mode combo-box and press the button close to Instance type textbox. Then, we double click on the t3.micro instance type in the pop-up window that contains the instance type list. Now, t3.micro is in the Instance type textbox. The rest of the parameters are set to their default values. Finally, we press the Execute button:

A pop-up window will display the run log:

When the cluster is started, the infrastructure software installation is submitted.

Download the transcriptome, evaluation and annotation files

In this step, we are going to download the transcriptome generated by Trinity and the filtered and clustered transcriptome, the complete annotation file, and the result files yielded by RSEM-EVAL. Due to the size of the transcriptome file, we are going to compress it previously.

To compress the assembly file, we first create the configuration file by selecting the menu item with the following path:

Main menu > Datasets > Result dataset file compression/decompression > Recreate config file

In the displayed window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box, **PcanCIC** in the *Experiment id* combo-box, **uncompressed** in the *Dataset status* combo-box (because the dataset has not been previously compressed), **trinity-170929-104827** in the *Result dataset* combo-box, and we type **Trinity.fasta.gz** as the pattern to select the files in the *File pattern* textbox and the local directory where the files will be downloaded in the *Local directory* textbox (or we select it using the button close to the textbox). Then we press the *Execute* button:

In the pop-up window, we can examine the config file. In this example, it has three sections: *identification*, with the dataset type and the experiment and dataset identifications, the action to do; and *file-1* with the file name of the Trinity assembly (in this example, we have only selected this file):

To compress the assembly file, we select the menu item with this path:

Main menu > Dataset > Result dataset compression/decompression > Run compression/decompression process

In the displayed window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the submission log:

We can view the process log during and after its run. To do so, we select the menu item with this path:

Main menu > Logs > View result logs in the cluster

In the raised window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box and **PcanCIC** in the *Experiment id* combo-box; and then we press the *Execute* button:

A pop-up window will display the results datasets for each run of the bioinformatic programs that correspond to the experiment:

Experiment id	Result dataset	Bioinfo app / Utility	Date	Time
PcanCIC	cdhitest-171013-115617	CD-HIT-EST	2017-10-13	11:56:17
PcanCIC	fastqc-170925-172321	FastQC	2017-09-25	17:23:21
PcanCIC	gzip-171114-155633	gzip	2017-11-14	15:56:33
PcanCIC	rsemeval-170930-101146	RSEM-EVAL	2017-09-30	10:11:46
PcanCIC	transbastx-171114-133352	transcriptome-blastx	2017-11-14	13:33:52
PcanCIC	transfil-171011-123558	transcript-filter	2017-10-11	12:35:58
PcanCIC	trimmo-170927-202434	Trimmomatic	2017-09-27	20:24:34
PcanCIC	trinity-170929-104827	Trinity	2017-09-29	10:48:27

Five results datasets are shown. We click on the **gzip-171114-155633** row, the dataset generated by gzip, the compression program, and another pop-up window will display its corresponding log:

Next, we are going to download the compressed assembly file. To do so, we first create the configuration file by selecting the menu item with the following path:

Main menu > Datasets > Result dataset file transfer > Recreate config file

In the raised window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box, **PcanCIC** in the *Experiment id* combo-box, **uncompressed** in the *Dataset status* combo-box (because the dataset has not been previously compressed), **trinity-170929-104827** in the *Result dataset* combo-box, and we type **Trinity.fasta.gz** as the pattern to select the files in the *File pattern* textbox and the local directory where the files will be downloaded in the *Local directory* textbox (or we select it using the button close to the textbox). Then we press the *Execute* button:

In the pop-up window, we can examine the config file. In this example, it has two sections: *identification*, with the experiment and results dataset identifications, the status of the dataset and the local path where files will be download; and *file-1* with the file name of the compressed Trinity assembly (in this example, we have only selected this file):

To download the compressed Trinity assembly from the cluster, we select the menu item with this path:

Main menu > Dataset > Result dataset file transfer > Download dataset from a cluster

In the displayed window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the log corresponding to the download:

Next, we are going to download the assembly quality assessment files generated by RSEM-EVAL. We have to download the ".results" files generated by this program. To do so, we first create the configuration file by selecting the menu item with the following path:

Main menu > Datasets > Result dataset file transfer > Recreate config file

In the raised window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box, **PcanCIC** in the *Experiment id* combo-box, **uncompressed** in the *Status* combo-box (because the dataset has not been previously compressed), **rsemeval-170930-101146** in the *Result dataset* combo-box, and we type **.*results** as the pattern to select the files in the *File pattern* textbox and the local directory where the files will be downloaded in the *Local directory* textbox (or we select it using the button close to the textbox). Then we press the *Execute* button:

In the pop-up window, we can examine the config file. In this example, it has five sections: *identification*, with the experiment and results dataset identifications, the status of the dataset and the local path where files will be downloaded; and *file-1* to *file-4* with the file names of four results files:

To download the results files from the cluster, we select the menu item with this path:

Main menu > Dataset > Result dataset file transfer > Download dataset from a cluster

In the displayed window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the log corresponding to the download:

And finally, we are going to download the annotation file generated by transcriptome-blastx. We have to download the file annotation.xml generated by this program. To do so, we first create the configuration file by selecting the menu item with the following path:

Main menu > Datasets > Result dataset file transfer > Recreate config file

In the displayed window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box, **PcanCIC** in the *Experiment id* combo-box, **uncompressed** in the *Dataset status* combo-box (because the dataset has not been previously compressed), **transbastx-171114-133353** in the *Result dataset* combo-box, and we type **annotation.xml** as the pattern to select the files in the *File pattern* textbox and the local directory where the files will be downloaded in the *Local directory* textbox (or we select it using the button close to the textbox). Then we press the *Execute* button:

In the pop-up window, we can examine the config file. In this example, it has two sections: *identification*, with the experiment and results dataset identifications, the status of the dataset and the local path where files will be downloaded; and *file-1* with the file name of the annotation file:

To download the results files from the cluster, we select the menu item with this path:

Main menu > Dataset > Result dataset file transfer > Download dataset from a cluster

In the raised window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the log corresponding to the download:

Terminate the cluster with t3.micro instance type

Once the analysis is complete, we terminate the cluster PcanCIC-t3.micro. To do so, we select the menu item with this path:

Main menu > Cloud control > Cluster operation > Terminate cluster

In the displayed window, we select **PcanCIC-t3.micro** in the *Cluster name* combo-box; and then we press the *Execute* button:

A pop-up window will display the run log:

Use of the M4, C4 and R4 instance types in StarCluster mode

NGScloud2 in StarCluster mode uses the software package StarCluster to implement the cluster concept where we can connect several instances that work together in order to distribute scripts runs that access the same datasets.

But since the current version of StarCluster (0.95.6), which uses M3, C3 and R3 instance types, AWS offers new instance types:

- M4 and M5 general purpose ones
- C4 and C5 computed optimized ones
- R4 and R5 memory optimized ones.

These instance types are not supported by StarCluster.

Inasmuch as StarCluster is an open source software, we make available a workaround to be able to use M4, C4 and R4 instance types without any commitment on our part until a new version of StarCluster supports them. It consists of replacing a Python script with another in which we have added the new instance types (see below the instruction to Ubuntu Linux and Windows).

Once this workaround is applied, the M4, C4 and R4 can be used in mode "StarCluster". If you select the menu item with this path:

Main menu > Cloud control > Configuration > List instance types

In the raised window, you can see the characteristics of the instances type. The column **StarCluster** shows whether the instance type is supported natively, non-supported, or if it can be used as a consequence of the workaround application.

NGScloud2 - List instance types - Table

Use	Instance type	vCPUs	Memory (GiB	Processor	Clock speed	Nitro Syste	StarCluster	Generation
general purpose	t2.micro	1	1	Intel Xeon Scalable	up to 3.3 GHz	no	native	current
general purpose	t2.small	1	2	Intel Xeon Scalable	up to 3.3 GHz	no	native	current
general purpose	t2.medium	2	4	Intel Xeon Scalable	up to 3.3 GHz	no	native	current
general purpose	t2.large	2	8	Intel Xeon Scalable	up to 3.0 GHz	no	workaround	current
general purpose	t2.xlarge	4	16	Intel Xeon Scalable	up to 3.0 GHz	no	workaround	current
general purpose	t2.2xlarge	8	32	Intel Xeon Scalable	up to 3.0 GHz	no	workaround	current
general purpose	t3.micro	2	1	Intel Xeon Scalable	2.5 GHz	yes	non-supported	current
general purpose	t3.small	2	2	Intel Xeon Scalable	2.5 GHz	yes	non-supported	current
general purpose	t3.medium	2	4	Intel Xeon Scalable	2.5 GHz	yes	non-supported	current
general purpose	t3.large	2	8	Intel Xeon Scalable	2.5 GHz	yes	non-supported	current
general purpose	t3.xlarge	4	16	Intel Xeon Scalable	2.5 GHz	yes	non-supported	current
general purpose	t3.2xlarge	8	32	Intel Xeon Scalable	2.5 GHz	yes	non-supported	current
general purpose	t3a.micro	2	1	AMD EPYC 7000 serie	2.5 GHz	yes	non-supported	current
general purpose	t3a.small	2	2	AMD EPYC 7000 serie	2.5 GHz	yes	non-supported	current
general purpose	t3a.medium	2	4	AMD EPYC 7000 serie	2.5 GHz	yes	non-supported	current
general purpose	t3a.large	2	8	AMD EPYC 7000 serie	2.5 GHz	yes	non-supported	current
general purpose	t3a.xlarge	4	16	AMD EPYC 7000 serie	2.5 GHz	yes	non-supported	current

Application of the workaround

Below the instructions to apply the workaround in Ubuntu Linux and Windows computers are described.

Ubuntu Linux and Mac OS X

First, open a terminal window and set the directory where the NGScloud2 package is installed (e.g. NGScloud2_package_path) as working directory and edit the file sci.sh typing the following commands:

```
$ cd NGScloud2_package_path
```

```
$ nano sci.sh
```

You have to review the variable **PYTHON2PATH** to put the command path to execute Python 2. The original value is **python2** and it is very possible that this value has not to be modified. Then, run the sci.sh script:

```
$ ./sci.sh
```

The information of the package StarCluster will be shown. The head FILE indicates the absolute path of the script __init__py and the corresponding directory is where the StarCluser package is installed.

In this Ubuntu example, the directory is **/usr/local/lib/python2.7/dist-packages/starcluster**. Now, you have to copy the static.py located in the directory `NGScloud2_package_path\additional_instances_workaround` to the directory where is StarCluster. For this, type the following commands:

```
$ cd /usr/local/lib/python2.7/dist-packages/starcluster

$ sudo mv static.py static.py.old

$ sudo mv static.pyc static.pyc.old

$ sudo cp NGScloud2_package_path/additional_instances_workaround/static.py .

$ sudo chmod 644 static.py
```

Windows

First, a Command Prompt started as Administrator, open a set the directory where the NGScloud2 package is installed (e.g. NGScloud2_package_path) as working directory typing the command:

```
> cd NGScloud2_package_path
```

The program sci.py shows information of StarCluster package. The script sci.bat allows to execute sci.py to Microsoft Windows users calling the Python interpreter. You may have to modify the value of some environment variables in sci.bat for its proper work. Then, type:

```
> sci
```

The output with the information of the package StarCluster will be shown. The head FILE indicates the absolute path of the script __init__py and the corresponding directory is where StarCluser is installed.

In this Windows example, the directory is **c:\programdata\anaconda3\envs\py27\lib\site-packages\starcluster**. Now, you have to copy the static.py located in the directory NGScloud2_package_path\additional_instances_workaround to the directory where StarCluster is. For this, type the following commands:

```
> cd c:\programdata\anaconda3\envs\py27\lib\site-packages\starcluster

> ren static.py static.py.old

> ren static.pyc static.pyc.old

> copy NGScloud2_package_path\additional_instances_workaround\static.py
```

Emergency procedure to terminate a cluster

If there are problems when you are creating or terminating a cluster (Internet connection, time out, etc.) and the cluster cannot be terminated in the usual way, there is an emergency procedure by selecting the menu item with this path:

Main menu > Cloud control > Cluster operation > Force termination of a cluster

In the raised window, we double click on the instance type that we need to terminate, e.g. r5.xlarge, in the pop-up window that contains the instance type list. Now, **r5.xlarge** is in the *Instance type* textbox. Finally, we press the *Execute* button:

A pop-up window will display the run log:

NGScloud - Cluster operation - Force termination of a cluster - Log
This process might take a few minutes. Do not close this window, please wait!
****
Verifying process requirements ...
Process requirements are OK.
****
Terminating the cluster PcanCIC-r3.xlarge using StarCluster ...
StarCluster - (http://star.mit.edu/cluster) (v. 0.95.6)
Software Tools for Academics and Researchers (STAR)
Please submit bug reports to starcluster@mit.edu
*** WARNING - Ignoring cluster settings due to --force option
>>> Terminating i-0941c93225d751e16
>>> Running plugin starcluster.plugins.sge.SGEPlugin
>>> Running plugin starcluster.clustersetup.DefaultClusterSetup
>>> Waiting for cluster to terminate...
>>> Removing placement group: @sc-PcanCIC-r3.xlarge
>>> Removing security group: @sc-PcanCIC-r3.xlarge
The cluster is terminated.
****
You can close this window now.

Enlarging a volume

Currently, NGSCloud2 has not an automatic utility to enlarge a EBS volume, but the enlarging can be done manually performing the following steps:

1. Create a new volume with the new size in the menu item with this path:

   Main menu > Cloud control > Volume Operation > Create volume

2. Create a cluster If there is not a cluster running in the menu item with this path:

   Main menu > Cloud control > Cluster operation > Create cluster

   In the raised window, we select native in the Cluster mode combo-box and press the button close to Instance type textbox. Then, we double click on the **t3.medium** instance type in the pop-up window that contains the instance type list. Now, **t3.medium** is in the Instance type textbox. The rest of the parameters keep the default value:

   WARNING: Do not select **spot** in Purchasing option combo-box because a spot instance can be stopped by AWS if it is required.
   Finally, we press the Execute button.

3. Mount the new volume in the cluster in the menu item with this path:

   Main menu > Cloud control > Volume operation > Mount volume in a node

In the raised window, we select the create cluster (**PcanCIC-t3.medium** in this example) in the Cluster name combo-box, **instance** in the Node name combo.box and the new volume (**PcanCIC-reads2** in this example) in the volume name combo-box, and type **/dev/sdm** in Device file textbox and **/change** in Mount path texbox:

Finally, we press the Execute button. A pop-up window will display the run log:

4. Open a terminal of the instance in the menu item with this path:

   Main menu > Cloud control > Open a terminal

In the raised window, we select the create cluster (**PcanCIC-t3.medium** in this example) in the Cluster name combo-box and **instance** in the Node name combo.box:

Then press the Execute button and a terminal will be opened:

5. If the dataset structure is single volume, type the following command in the terminal:

   ```
   cp -r /ngscloud2/* /change
   ```

   If the dataset structure is multivolume, type the following command in the terminal:

   `cp -r /ngscloud2/`**dataset-type**`/* /change`

   where dataset-type can be: **apps**, **databases**, **reads**, **references** and **results**. E.g. if the volume to enlarge is where reads are stored, the command will be:

   ```
   cp -r /ngscloud2/reads/* /change
   ```

6. Relink the new volume in the menu item with this path:

   Main menu > Cloud control > Configuration > Link volumes

7. Terminate the cluster in the menu item with this path:

   Main menu > Cloud control > Cluster operation > Terminate cluster

8. Create a new cluster in the menu item with this path:

   Main menu > Cloud control > Cluster operation > Create cluster

9. Check data of the new volume linked in the menu item with this path:

   Datasets > List dataset

10. If the new volume linked is **OK**, the old volume can be removed in the menu item with this path:

    Main menu > Cloud control > Volume Operation > Remove volume

    WARNING: Be careful! First check that the new volume data are OK (step 9) because this action cannot be undone.

How-to...

Below is the menu path of the tasks the most common tasks:

How to display this manual

Main menu > Help > View help ... or pressing F1 key

How to recreate the NGScloud2 config file

Main menu > Cloud control > Configuration > Recreate NGScloud2 config file

How to create a new environment

Main menu > Cloud control > Set environment

How to change to another environment

Main menu > Cloud control > Set environment

How to view characteristics of the instance types

Main menu > Cloud control > Configuration > List instance types

How to create a cluster

Main menu > Cloud control > Cluster operation > Create cluster

How to terminate a cluster

Main menu > Cloud control > Cluster operation > Terminate cluster

If you have problems when you are creating or terminating a cluster, there is an emergency procedure:

Main menu > Cloud control > Cluster operation > Force termination of a cluster

How to list the running clusters

Main menu > Cloud control > Cluster operation > List clusters

How to create a volume

Main menu > Cloud control > Volume Operation > Create volume

How to remove a volume

Main menu > Cloud control > Volume Operation > Remove volume

How to list the created volumes

Main menu > Cloud control > Volume Operation > List volumes

How to links volumes in the environment configuration

Main menu > Cloud control > Configuration > Link volumes

How to add a node in a cluster (only StarCluster cluster mode)

Main menu > Cloud control > Node operation > Add node in a cluster

How to remove a node in a cluster (only StarCluster cluster mode)

Main menu > Cloud control > Node operation > Remove node in a cluster

How to open a terminal of a cluster

Main menu > Cloud control > Open a terminal

How to set up a bioinformatic software in a cluster

Main menu > Cloud control > Bioinfo software install > bioinformatic software to use

How to run a *De novo* RNA-seq software in a cluster

Main menu > De novo RNA-seq > "Task " > "Bioinformatic software" > Recreate config file

Main menu > De novo RNA-seq > "Task " > "Bioinformatic software" > "Run process"

Task	***Bioinformatic software***
Read quality	FastQC
Trimming	cutadapt
	Trimmomatic
Digital normalization	Insilico_read_normalization (Trinity package)
Assembly	SOAPdenovo-Trans
	Trans-ABySS
	Trinity
Read alignment	Bowtie2
	GSNAP (GMAP-GSNAP package)
Transcriptome quality assessment	BUSCO
	QUAST
	rnaQUAST
	RSEM-EVAL (DETONATE package)
	Transrate
Transcriptome filtering	CD-HIT-EST (CD-HIT package)
	transcript-filter (NGShelper package)
Quantitation	eXpress
	kallisto
Variant calling	Variant calling
Annotation	Transcriptome-blast (NGShelper package)

How to run a Reference-based RNA-seq software in a cluster

Main menu > Reference-based RNA-seq > "Task " > "Bioinformatic software" > Recreate config file

Main menu > Reference-based RNA-seq > "Task " > "Bioinformatic software" > "Run process"

Task	***Bioinformatic software***
Read quality	FastQC
Trimming	cutadapt
	Trimmomatic
Assembly	Cufflinks-Ciffmerge (Cufflinks package)
	Genome-guided Trinity (Trinity package)
Read alignment	Bowtie2
	GSNAP (GMAP-GSNAP package)
	HISAT2
	STAR
	TopHat

Transcriptome alignment	GMAP (GMAP-GSNAP package)
Quantitation	Cuffquant (Cufflinks package)
	htseq-count (HTSeq package)
Differential expression	Cuffdiff (Cufflinks package)
	Cuffnorm (Cufflinks package)
Variant calling	Variant calling

How to run a RAd-seq software in a cluster

Main menu > RAd-seq > "Task " > "Bioinformatic software" > Recreate config file

Main menu > RAD -seq > "Task " > "Bioinformatic software" > "Run process"

Task	***Bioinformatic software***
Enzyme analysis	rsitesearch (ddRADseqTools package)
In silico simulations	ddRADseq simulation (ddRADseqTools package)
RAD design	RADdesigner
Read quality	FastQC
Trimming	cutadapt
	Trimmomatic
Pseudo assembly	SOAPdenovo2
	starcode
Read alignment	Bowtie2
	GSNAP (GMAP-GSNAP package)
	HISAT2
Transcriptome alignment	GMAP (GMAP-GSNAP package)
Variant calling	Variant calling
Pipelines	ipyrad

How to display datasets of a volume

Main menu > Dataset > List dataset

How to display the contents of a dataset

Main menu > Dataset > List dataset

How to upload reference files to a cluster

Main menu > Datasets > Reference dataset file transfer > Recreate config file

Main menu > Datasets > Reference dataset file transfer > Upload dataset to a cluster

How to compress/decompress reference files in a cluster

Main menu > Datasets > Reference dataset file compression/decompression > Recreate config file

Main menu > Datasets > Reference dataset file compression/decompression > Run compression/decompression process

How to upload database files to a cluster

Main menu > Datasets > Database dataset file transfer > Recreate config file

Main menu > Datasets > Database dataset file transfer > Upload dataset to a cluster

How to compress/decompress database files in a cluster

Main menu > Datasets > Database dataset file compression/decompression > Recreate config file

Main menu > Datasets > Database dataset file compression/decompression > Run compression/decompression process

How to upload read files to a cluster

Main menu > Datasets > Read dataset file transfer > Recreate config file

Main menu > Datasets > Read dataset file transfer > Upload dataset to a cluster

How to compress/decompress read files in a cluster

Main menu > Datasets > Read dataset file compression/decompression > Recreate config file

Main menu > Datasets > Read dataset file compression/decompression > Run compression/decompression process

How to download results files from a cluster

Main menu > Datasets > Result dataset file transfer > Recreate config file

Main menu > Datasets > Result dataset file transfer > Download dataset from a cluster

How to compress/decompress results files in a cluster

Main menu > Datasets > Result dataset file compression/decompression > Recreate config file

Main menu > Datasets > Result dataset file compression/decompression > Run compression/decompression process

How to view submission logs in the local computer

Main menu > Logs > View submission logs in the local computer

How to view result logs in the cluster

Main menu > Logs > View result logs in the cluster

How to cite

If you are using NGScloud2, you should cite the following papers:

Mora-Márquez, F., Vázquez-Poletti, J. L., & López de Heredia, U. (2018). NGScloud: RNA-seq analysis of non-model species using cloud computing. *Bioinformatics*, *34*(19), 3405–3407. doi:10.1093/bioinformatics/bty363

Mora-Márquez, F., Vázquez-Poletti, J. L., & López de Heredia, U. (2020). NGScloud2: optimized bioinformatic analysis using Amazon Web Services. *bioRxiv* 2020.11.30.400010. doi: 10.1101/2020.11.30.400010