Nextflow Overview

Nextflow is a workflow management system designed for organizing and automating bioinformatics analyses. It allows you to define **processes** (e.g., data pre-processing, alignment, quantification) and connect them using **channels** to pass data between steps. Together, these form a **workflow** that automates your entire analysis pipeline.

Key Features

- **Parallelization**: Run multiple tasks simultaneously, speeding up analyses using multiple CPU cores or distributed systems.
- **Re-entrancy**: Resume interrupted workflows from where they stopped using the `-resume` flag, avoiding redundant reprocessing.
- **Reusability**: Easily adapt workflows for different datasets or environments. For instance, the nf-core RNA-seq pipeline offers a pre-configured, community-driven solution for RNA-seq analysis.

By understanding these features, you can create efficient and reliable pipelines for bioinformatics tasks.

Hoffman2 Cluster at UCLA

The **Hoffman2 Cluster** is a high-performance computing system for large-scale computational tasks. It supports researchers working with massive datasets or complex simulations that exceed the capabilities of standard computers.

What is a Cluster?

A cluster is a network of connected computers (nodes) working together as a single system. This setup allows for faster processing and handling of large or complex tasks, such as genomics analyses or simulations.

About Hoffman2

Hoffman2 is UCLA's shared HPC system, ideal for resource-intensive projects. It features:

- Over 25,000 CPU cores across 800+ nodes
- 174TB of memory and 8PB of network storage
- General-use and condo-style sections (for priority access with purchased nodes)

Applying for an Account

To request a Hoffman2 account, follow the instructions at: Request an Account. After applying, check your status at: Account Status.

Accessing Hoffman2

Prerequisites

- UCLA network connection or VPN
- Hoffman2 username and password

Login Steps

1. Open the terminal on your computer.
2. Log in using Secure Shell (SSH) with the command:
   ssh username@hoffman2.idre.ucla.edu
3. Enter your password when prompted.

```
ssh your_username@hoffman2.idre.ucla.edu
```

> Replace `your_username` with your actual Hoffman2 username
>
> Note: Use MobaXterm for Windows PC users

You'll be prompted to enter your password after entering the above command into your terminal.

Troubleshooting

- If you can't connect, verify you are on the UCLA network or using a VPN.
- Ensure you are entering the correct username.
- Contact Hoffman2 Support if issues persist.

Nextflow Installation and Setup

1. **Scratch Space**: `/u/scratch/your_initial/your_username/`
    - 2TB of temporary storage
    - Data is wiped at the end of every quarter
2. **Paid Project Space for Allard Lab**: `/u/project/pallard/your_username/`
    - 1TB of permanent storage
    - Data is not wiped
    - Ideal for storing raw sequencing data files

Spinning up an interactive node

```
# To spin up a node
qrsh -l h_data=15G,h_rt=3:00:00,h_vmem=4G -pe shared 4
```

Installing JAVA

```
# Loading java module
module load java/jdk-17.0.12

#install a wrapper for java
curl -s https://get.sdkman.io | bash
```

Confirm Java version/Installation:

```
java -version
```

Install Nextflow and make it executable:

```
#Install Nextflow
curl -fsSL https://get.nextflow.io | bash

#Granting permissions
chmod +x nextflow
```

Move nextflow into your `HOME` directory and export to `PATH`

```
# Creates a bin directory if you don't have one
mkdir -p $HOME/.local/bin/
# Moves nextflow into the bin directory
mv nextflow $HOME/.local/bin/

# To check directory creation
cd $HOME/.local/bin/

# List directory contents
ls

# Exports path
export PATH="$HOME/nextflow:$PATH"

# Takes you back to your home directory
cd
```

Run the following lines of code to reset your `.bashrc` file. Then, follow the steps below to make your `.bashrc` file

```
# Restore
/bin/cp /etc/skel/.bashrc ~/

# Source
source ~/.bashrc
```

Creating a bash profile with nano (has all the info required to access and run Nextflow)

```
nano ~/.bashrc
```

Instructions for Adding Information to `.bashrc`

1. **Copy Code to a Text Editor First:**
    - **Do not** copy the code directly from these instructions and paste it into the terminal while using `nano`. Doing so can cause unintended changes to the code's formatting and indentation.
    - Instead, copy the information into a text editor like **Sublime Text**, **Notepad**, or the macOS **TextEdit** application.
    - For single-line commands (no indentation), copying directly is acceptable.
2. **Check and Correct the Indentation:**
    - Paste the copied content into your text editor.

    - Verify that the indentation is correct and there are no unwanted line breaks or spaces.
3. **Open `.bashrc` for Editing:**
    - Open your `.bashrc` file using the `nano` editor (or another terminal editor):
4. **Paste the Information into `.bashrc`:**
    - Once the file is open in `nano`, paste the content from your text editor into the file.
5. **Save and Exit:**
    - Save the changes by pressing `Ctrl+O`, then press `Enter`.
    - Exit the editor by pressing `Ctrl+X`.
6. **Apply the Changes:**
    - Run the following command to source the `.bashrc` file and apply the changes: source ~/.bashrc

Important:

- Do not copy the code from here directly.
- Refer to the information posted on Slack for the correct formatting.
- Copy the code into a text editor to verify formatting before transferring it to the `.bashrc` file.

Paths and Modules to Add to Your `.bashrc` File

```
# Nextflow and HPC Environment Configuration
export PATH="$HOME/.local/bin:$PATH"
export NXF_SINGULARITY_CACHEDIR="${SCRATCH:-/tmp}/cache"
export PATH="$HOME/nextflow:$PATH"

# Set the PATH to include Java, Intel Compiler, and other system
directories
export
PATH="$HOME/.sdkman/candidates/java/current/bin:$HOME/.local/compilers/inte
l/2020.4/compilers_and_libraries_2020.4.304/linux/bin/intel64:$HOME/.local/
compilers/intel/2020.4/compilers_and_libraries_2020.4.304/linux/bin:$HOME/.
local/compilers/intel/2020.4/compilers_and_libraries_2020.4.304/linux/mpi/i
ntel64/libfabric/bin:$HOME/.local/compilers/intel/2020.4/compilers_and_libr
aries_2020.4.304/linux/mpi/intel64/bin:$HOME/.local/compilers/intel/2020.4/
debugger_2020/gdb/intel64/bin:/u/systems/UGE8.6.4/bin/lx-amd64:$HOME/.local
/bin:$PATH"

# Loading modules
module load python/3.9.6
module load singularity/3.8.5
module load java/jdk-17.0.12
```

```
module load java/jdk-11.0.14

# Add custom directories to PATH
export PATH="$HOME/.local/bin:$PATH"

# Add specific tool directories if they are not in .local/bin
if [[ ":$PATH:" != *":$HOME/.local/bin:"* ]]; then
    export PATH="$HOME/.local/bin:$PATH"
fi
```

Save your `.bashrc`
- `Ctrl + O` to save
- `Ctrl + X` to quit (press Y to save changes)

Source your .bashrc

```
source ~/.bashrc
```

Confirm that Nextflow is installed correctly:

```
nextflow info
```

Installing nf-core Pipelines

nf-core provides a collection of community-curated Nextflow pipelines designed for reproducible and scalable bioinformatics workflows. These pipelines cover various domains, including RNA sequencing (RNA-seq), DNA methylation analysis, variant calling, and more. To utilize these pipelines, they must first be pulled from the nf-core GitHub repository.

Prerequisites

Before installing nf-core pipelines, ensure that you have the following installed on your system:

- **Nextflow**: A workflow management system for scalable and reproducible scientific workflows.
- **Java**: Required for running Nextflow. HPC environments usually have java modules pre-installed which can be accessed by users any time.
- **Docker or Singularity**: Enables containerized execution of pipelines for consistency across different systems. HPC environments usually have either one of the tools installed already.
- **Git (Optional)**: Useful for cloning repositories and managing versions of nf-core pipelines.

Pulling nf-core Pipelines

The `nextflow pull` command allows users to retrieve and locally cache a copy of an nf-core pipeline. This ensures that the workflow and its dependencies are available for execution.

1. Pulling the RNA-seq Pipeline

To download the latest version of the **RNA-seq analysis pipeline**, run:

```
nextflow pull nf-core/rnaseq
```

This command fetches the RNA-seq pipeline from the nf-core GitHub repository and caches it in the Nextflow home directory (typically `~/.nextflow/assets/nf-core/`).

2. Pulling the Methylation Sequencing Pipeline

To download the **methylation sequencing (methylseq) pipeline**, use:

```
nextflow pull nf-core/methylseq
```

This pipeline is designed for analyzing bisulfite sequencing data, including Whole Genome Bisulfite Sequencing (WGBS) and Reduced Representation Bisulfite Sequencing (RRBS).

Verifying Installation

Once the pipelines are pulled, verify their availability by listing cached pipelines:

```
nextflow list
```

To check for updates or new releases of a pipeline, use:

```
nextflow pull nf-core/<pipeline_name> -r <version>
```

For more information on configuring and customizing nf-core pipelines, refer to the official nf-core documentation.

Installing `mksquashfs`

In Nextflow workflows, especially in distributed file systems or high-performance computing (HPC) environments, `mksquashfs` is used to create **SquashFS** images. SquashFS is a compressed, read-only file system that bundles files and directories into a single image file.

This tool is commonly employed in Nextflow workflows to overcome data distribution challenges and optimize containerized execution in distributed environments. Distributed file systems often experience performance bottlenecks when workflows require frequent access to many small files. A SquashFS image addresses this issue by combining files into a single archive, reducing the number of I/O operations and improving efficiency.

Steps to Install `mksquashfs`

1. **Ensure Node Access:**
    - Before proceeding, ensure that you have spun up a node for installation.
2. **Prepare Code for Execution:**
    - Copy the installation commands into a text editor like **Sublime Text** or **Notepad** before pasting them into the terminal.
    - This ensures the commands are formatted correctly and prevents errors during execution.
3. **Run the Installation Commands:**
    - Paste the prepared commands into the terminal and execute them to install `mksquashfs`.

Make sure to spin up a node if you are not on a node already!

```
qrsh -l h_data=15G,h_rt=3:00:00,h_vmem=4G -pe shared 4
```

Installation:

```
# Navigate to the target directory and download squashfs source
cd $SCRATCH
wget
https://sourceforge.net/projects/squashfs/files/squashfs/squashfs4.6.1/squa
shfs4.6.1.tar.gz/download -O squashfs4.6.1.tar.gz

# Extract the downloaded tarball
tar -xvzf squashfs4.6.1.tar.gz

# Navigate to the source folder
cd squashfs-tools-4.6.1/squashfs-tools/

# Compile the squashfs-tools
make
```

```
# Verify the build outputs
ls

# Copy the compiled binaries to local bin directories
cp mksquashfs unsquashfs ~/.local/bin

# Source bashrc
source ~/.bashrc

# Test the installation
which mksquashfs
```

Uploading and Accessing Files Through FileZilla

Prerequisites

- FileZilla installed on your computer
- Active UCLA campus network connection or VPN
- Your Hoffman2 credentials

Setup Steps

1. **Open FileZilla:** Launch the application.
2. **Access Site Manager:** Click on `File → Site Manager` (or press `Ctrl+S`).
3. **Create a New Site:**
   - Click `New Site` and name it "Hoffman2."
4. **Enter Connection Details:**
   - **Host:** `hoffman2.idre.ucla.edu`
   - **Port:** `22`
   - **Protocol:** `SFTP - SSH File Transfer Protocol`
   - **Logon Type:** `Normal`
   - **User:** `your_username`
   - **Password:** `your_password`

   Important: Replace `your_username` and `your_password` with your actual Hoffman2 credentials.

5. **Save and Connect:** Click `Connect` to save the settings and connect to Hoffman2.

Using FileZilla

- **Left Panel:** Displays your local computer files.
- **Right Panel:** Displays Hoffman2 remote files.
- **To Transfer Files:**
    - Drag and drop files between the panels.
    - Or right-click on the file and select "Upload" or "Download."

Navigating to Directories

Under the `Remote site` option in FileZilla, enter the directory path where the FASTQ files are located to navigate directly to the desired folder.

```
/u/project/pallard/sujit009/rnaseq_20241111/escs_epilcs
```

Sample Sheet Preparation

List the paths of all the FASTQ files. We will use this information to make our sample sheet whill will be used by the Nextflow pipeline to map our files.

```
find /u/project/pallard/sujit009/rnaseq_20241111/escs_epilcs -type f -name "*.fastq.gz"
```

Create a CSV file with the following columns:

- **sample**: Unique sample identifier
- **fastq_1**: Path to the first FASTQ file
- **fastq_2**: Path to the second FASTQ file (for paired-end data)
- **strandedness**: RNA-seq strandedness (auto/forward/reverse/unstranded)

Example sample sheet:

sample,fastq_1,fastq_2,strandedness
sample1,/path/to/sample1_R1.fastq.gz,/path/to/sample1_R2.fastq.gz,auto
sample2,/path/to/sample2_R1.fastq.gz,/path/to/sample2_R2.fastq.gz,auto
sample3,/path/to/sample3_R1.fastq.gz,/path/to/sample3_R2.fastq.gz,auto

Tips:

- Save the file with a `.csv` extension
- Use full paths to FASTQ files
- Ensure all FASTQ files are accessible on Hoffman2
- If unsure about strandedness, use `auto`
- Double-check that there are no spaces after commas

Example sample sheet:

Note: Reads from multiple sequencing lanes are automatically concatenated if they share the same sample name in the sample sheet. The samples highlighted in yellow indicate instances where sequencing was performed across multiple lanes. By assigning the same sample name, Nextflow ensures that reads from all lanes are merged correctly during processing.

```
sample,fastq_1,fastq_2,strandedness
ESCs0uMREP1,/u/project/pallard/sujit009/rnaseq_20241111/escs_epilcs/ESCs-0nM-2_S25_L003_R1_001.fastq.gz,/u/project/pallard/sujit009/rnaseq_20241111/escs_epilcs/ESCs-0nM-2_S25_L003_R2_001.fastq.gz,auto
ESCs0uMREP1,/u/project/pallard/sujit009/rnaseq_20241111/escs_epilcs/ESCs-0nM-2_S25_L004_R1_001.fastq.gz,/u/project/pallard/sujit009/rnaseq_20241111/escs_epilcs/ESCs-0nM-2_S25_L004_R2_001.fastq.gz,auto
ESCs0uMREP2,/u/project/pallard/sujit009/rnaseq_20241111/escs_epilcs/ESCs-0nM-3_S28_L003_R1_001.fastq.gz,/u/project/pallard/sujit009/rnaseq_20241111/escs_epilcs/ESCs-0nM-3_S28_L003_R2_001.fastq.gz,auto
ESCs0uMREP2,/u/project/pallard/sujit009/rnaseq_20241111/escs_epilcs/ESCs-0nM-3_S28_L004_R1_001.fastq.gz,/u/project/pallard/sujit009/rnaseq_20241111/escs_epilcs/ESCs-0nM-3_S28_L004_R2_001.fastq.gz,auto
ESCs0uMREP3,/u/project/pallard/sujit009/rnaseq_20241111/escs_epilcs/ESCs-0nM-4_S31_L003_R1_001.fastq.gz,/u/project/pallard/sujit009/rnaseq_20241111/escs_epilcs/ESCs-0nM-4_S31_L003_R2_001.fastq.gz,auto
ESCs0uMREP3,/u/project/pallard/sujit009/rnaseq_20241111/escs_epilcs/ESCs-0nM-4_S31_L004_R1_001.fastq.gz,/u/project/pallard/sujit009/rnaseq_20241111/escs_epilcs/ESCs-0nM-4_S31_L004_R2_001.fastq.gz,auto
```

Uploading to Hoffman2

1. **Create a Logical Directory Structure**
   Organize your data and files in a way that ensures clarity and efficiency. Note that data in the `scratch` space must be transferred to the lab server or a local hard disk **before the end of the quarter** to avoid data loss.

Example directory structure:

```
/u/scratch/your_initial/your_username/
└── your_experiment_name/
    ├── samplesheet.csv
    └── fastq/
```

2. **Using FileZilla to Upload Files**

1. Open **FileZilla** and connect to Hoffman2.

In the remote panel, navigate to your scratch directory:

```
/u/scratch/your_initial/your_username/
```

2. Create a new directory for your project:
   - Right-click → Create Directory
   - Name the directory meaningfully (e.g., `rnaseq_project_jan2024`).
3. Enter the new directory in the remote panel.
4. Upload your **samplesheet** and other required files:
   - In the local panel (left side), find `samplesheet.csv`.
   - Drag and drop it to the corresponding folder in the remote panel (right side).

Directory Structure Example

```
/u/scratch/your_initial/your_username/
└── rnaseq_project_jan2024/
    ├── samplesheet.csv
    └── fastq/
```

Generating the Bash Submission Script with the Helper HTML

Required Variables

Here are the paths you will need to include in your bash script:

```
SAMPLESHEET="/u/scratch/your_initial/your_username/your_experiment_name/samplesheet.csv"

OUTPUTDIR="/u/scratch/your_initial/your_username/your_experiment_name"

GENOME="/u/project/pallard/sujit009/genome/Mice/Mus_musculus.GRCm39.dna.primary_assembly.fa"

GTF="/u/project/pallard/sujit009/genome/Mice/Mus_musculus.GRCm39.113.gtf.gz"
```

Saving the Script

Save the bash script as a `.sh` file, for example:
`nextflow_rnaseq.sh`

Upload the bash script to your scratch directory:

```
/u/scratch/your_initial/your_username/
└── rnaseq_project_jan2024/
    ├── samplesheet.csv
    ├── nextflow_rnaseq.sh
```

Final Steps

1. Perform a sanity check:
    - Ensure both `samplesheet.csv` and `nextflow_rnaseq.sh` are present in the directory.
2. Once verified, submit the job through the terminal on Hoffman2.

```
cat
/u/scratch/your_initial/your_username/rnaseq_project_jan2024/samplesheet.cs
v
Cat
/u/scratch/your_initial/your_username/rnaseq_project_jan2024/nextflow_rnase
q.sh
```

Submitting Your Job

1. Submit your job to the cluster:

```
qsub
/u/scratch/your_initial/your_username/rnaseq_project_jan2024/nextflow_rnase
q.sh
```

2. Monitor your job status:

```
# View all your jobs
qstat -u $USER

# View detailed job information (replace JOBID)
qstat -j JOBID
```

Understanding Job Status

Common job states displayed in the `qstat` output:

- **`qw`**: Job is queued and waiting.
- **`r`**: Job is currently running.
- **`Eqw`**: Error in job submission; requires troubleshooting.
- **`t`**: Job is being transferred.
- **`d`**: Job is being deleted.

Managing Jobs

To manage your jobs, use the following commands:

Delete a specific job

```
qdel JOBID
```

Delete all your jobs

```
qdel JOBID
```

Best Practices

1. **Resource Requests**:
    - Request realistic **memory** and **time** limits to optimize queue efficiency.
    - Avoid overestimating resources, as this can increase queue times unnecessarily.
2. **Job Organization**:
    - Keep all scripts and logs **well-organized** for easier debugging and reference.
    - Use **meaningful job names** to identify tasks easily.
    - Document resource requirements for future use.
3. **Storage Management**:
    - Regularly clean up unnecessary files to free up space.

Monitor disk usage with:

```
du -h --max-depth=1
```

    - Use **scratch space** for storing temporary files and avoid cluttering home directories.