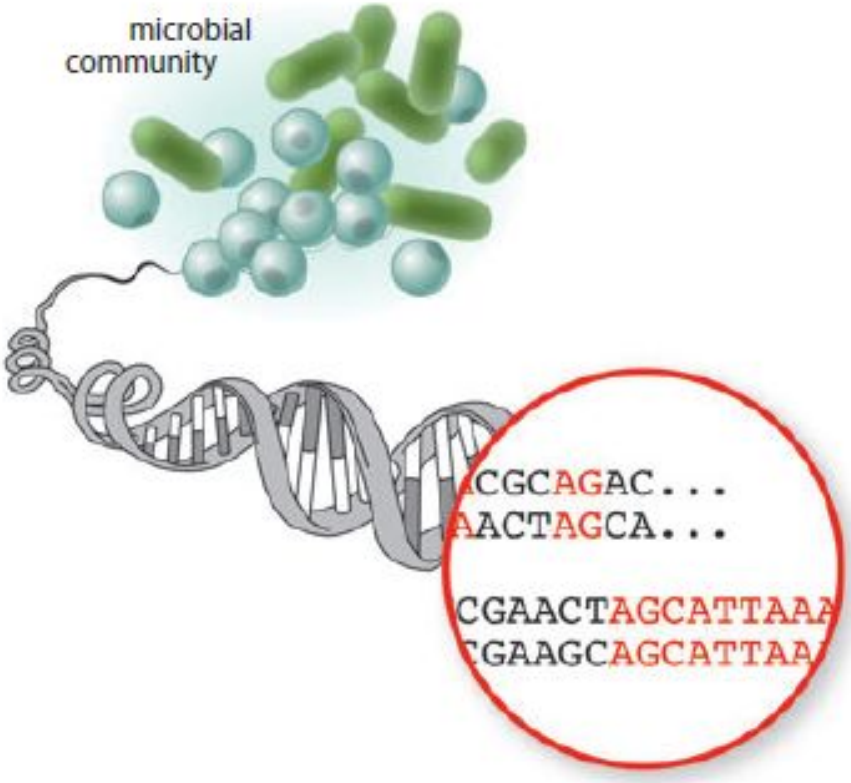
Rapport Projet Long

Implémentation d'un pipeline de métagénomique ciblée en Nextflow, et distribué via Docker

Etienne JEAN
Janvier 2019



Abréviations

ADN	Acide désoxyribonucléique
ARN	Acide ribonucléique
PCR	Amplification de fragments d'ADN (Polymerase Chain Reaction)
WGS	Séquençage de génome complet (Whole genome Sequencing)
Pipeline / Workflow	Chaîne de processus informatiques
Reads	Fragments d'ADN séquencés
ITS	Internal transcribed spacer
OTU	Operational Taxonomic Unit
single-end	Séquençage d'une seul extrémité d'un fragment
paired-end	Séquençage des deux extrémités d'un fragment
process	Processus informatique faisant partie d'un pipeline (Nextflow)
channel	Instance permettant le transfert de données d'un process à l'autre (Nextflow)
log	Fichier contenant un rapport d'exécution ou d'erreur
checkpoint	Sauvegarde de l'état du travail permettant la reprise après interruption

Introduction

La métagénomique est l'analyse génomique d'un ensemble de micro-organismes directement à partir de prélèvements d'échantillons. Etant donné que seule une mince proportion des micro-organismes vivants sont cultivables [5], cette technique permet l'étude de populations complexes et variées issues de différents milieux. La métagénomique ciblée est un sous domaine qui se concentre sur le séquençage de certains gènes particuliers après amplification par PCR, comme les sous-unités ribosomales, qui sont des séquences conservées dans la majorité des organismes vivants. Elle permet de mettre en lumière les différents micro-organismes présents dans l'échantillon. Cette technique reste moins exhaustive et précise que le séquençage de génomes complet (WGS) (elle ne permet pas la détection de certains organismes comme les virus), mais elle reste très utilisée pour son faible coût.

SHAMAN (SHiny Application for Metagenomics ANalysis) [3] est l'une des premières application web qui a permis aux biologistes d'effectuer des analyses quantitatives de données métagénomiques. SHAMAN utilise le workflow de métagénomique ciblée MASQUE (Metagenomic Analysis with a Quantitative pipeline) [3]. MASQUE permet l'analyse de données de séquençage d'ARN ribosomaux 16S et 23S issues de bactéries et d'archées, d'ARN 18S et 28S issues d'eucaryotes, et des ITS (Fig. 1). Les ARN 16S et 18S font partie de la petite sous-unité des ribosomes, et les ARN 23S et 28S de la grande sous-unité. Les ITS sont une région des gènes ribosomaux situés entre les séquences de la grande sous-unité et de la petite sous-unité ribosomales. Ils sont facilement amplifiés par PCR et sont particulièrements informatifs pour distinguer des espèces proches. Par annotation des clusters de reads séquencés (appelés OTU, pour Operational Taxonomic Unit), le pipeline permet de retrouver les espèces présentes dans l'échantillon et leur proportion.

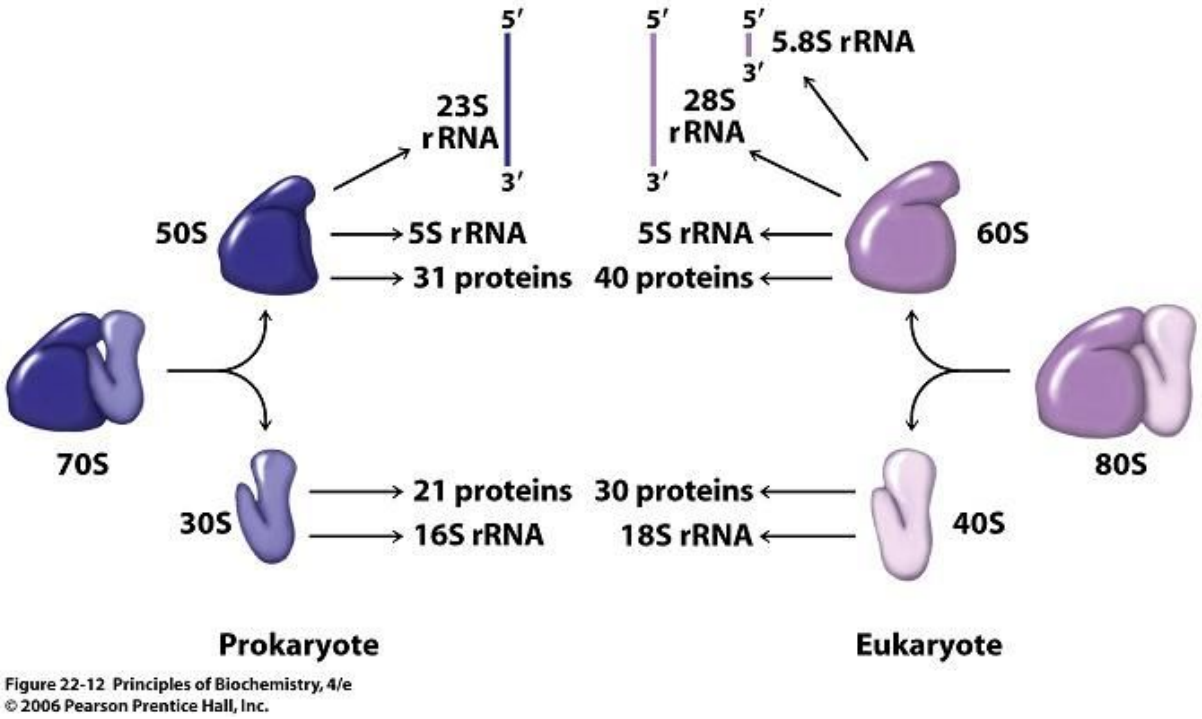


Figure 1 : *Composition des ribosomes chez les procaryotes et les eucaryotes. Le pipeline MASQUE utilise le séquençage de fragments d'ARN 16S, 18S, 23S, 28S et des ITS.*

Actuellement MASQUE est implémenté en bash. Mais bash n'est pas un langage de programmation conçu et adapté pour développer des workflow complexes. Le programme manque donc cruellement de flexibilité et rend la maintenance et les mises à jour laborieuses. De plus, le serveur web SHAMAN utilise MASQUE grâce aux ressources de calcul de l'Institut Pasteur, mais le déploiement de MASQUE sur d'autres types de supercalculateurs n'est pas forcément compatible en l'état, et peut demander des modifications du code source non triviales pour sa mise en fonctionnement.

Ainsi, le but de ce projet est de transcrire MASQUE en Nextflow, un langage spécifiquement développé pour implémenter des workflow, et qui réglerait les problèmes actuels de la version en bash, puis intégrer le programme dans l'image Docker de SHAMAN, afin qu'il puisse être finalement utilisé par l'application web.

Matériel et méthodes

MASQUE

Le code source de MASQUE est hébergé sur GitHub (https://github.com/aghozlane/masque). Ce script a été utilisé pour le développement du workflow de ce projet en Nextflow.

Nextflow

Nextflow est un *Domain-Specific Language*, un langage de programmation spécifiquement conçu pour le développement de workflow, récemment développé par le *Bioinformatics Comparative group* du Centre de Régulation Génomique (CRG) à Barcelone. Il est basé sur Groovy, un langage qui est un surcouche de Java. Nextflow fonctionne par enchaînements de *process*, qui effectuent les différentes tâches du pipeline, et connectés par des *channel*, qui transmettent les données.

Nextflow permet notamment la parallélisation automatique des process, la gestion poussée d'erreurs, avec génération de rapport d'erreur (logs) et l'introduction de checkpoints au cours de l'exécution, qui permettent de reprendre une exécution involontairement stoppée.

De plus, les process peuvent être gérés individuellement de manière bien plus avancée et aisée qu'en bash. Il est par exemple possible de gérer à la carte l'allocation des CPUs, des GPUs, et de la mémoire de manière dynamique, en fonction des ressources de l'exécuteur, ou encore de définir des règles de gestion d'erreurs spécifiques, d'imposer une limite de temps d'exécution, ou même d'associer une image Docker avec un process particulier.

Enfin, une puissance de Nextflow est sa capacité à gérer un workflow sur différents exécuteurs, notamment sur différents types de supercalculateurs ou sur les services de *cloud computing* comme *Amazon Web Service.* Il utilise pour ça la définition de certains paramètres dans un fichier de configuration séparé du code, ce qui permet la différenciation entre l'utilisateur et le développeur.

Docker

Docker est un programme qui permet l'exécution de tâches dans des systèmes virtuels appelés *conteneurs*. Le but de ce système est d'englober toutes les dépendances d'un programme dans une *image* (qui permet la génération de conteneurs) qui lui est associé, améliorant ainsi grandement sa portabilité d'une plateforme à une autre. Docker rend automatiquement compatible les conteneurs sur Windows, MacOS et Linux.

Ce système de conteneur est notamment bien plus optimisé que les machines virtuelles (Fig. 2). La mise en place d'un conteneur demande beaucoup moins de ressources informatiques et d'espace de stockage. D'ailleurs, Docker n'installe que la "différence" d'une nouvelle image par rapport aux images déjà préalablement téléchargées.

Le conteneur, qui est indépendant du système hôte de l'utilisateur, permet de se passer d'installations multiples, et donc d'éviter toute modification du système hôte, ainsi que d'avoir un programme fonctionnel en toutes circonstances.

Enfin, Docker propose l'hébergement web gratuit d'images sur le site *Docker Hub*. L'image Docker de SHAMAN, hébergé sur Docker Hub sous le nom *aghozlane/shaman*, a été utilisée comme image de base pour le développement de ce pipeline.

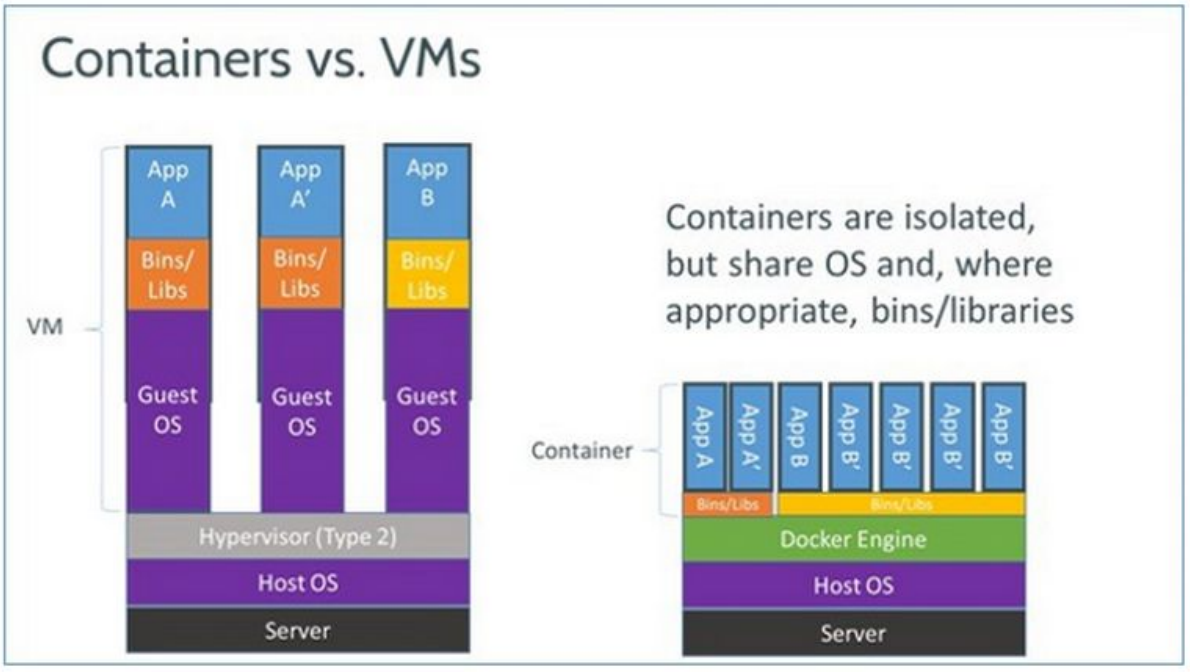


Figure 2 : *Comparaison des conteneurs Docker et des Machines virtuelles. Contrairement aux Machines virtuelles, Docker minimise l'allocation de ressources pour le fonctionnement des conteneurs en mutualisant le plus possible.*

Résultats

Le programme développé au cours de ce projet suit donc l'approche utilisée par MASQUE. Le code ainsi que des données tests sont hébergés sur Github (https://github.com/Etjean/Long_project) et son utilisation y est décrite.

Description du workflow (Fig. 4)

- **Données d'entrée.** Les reads obtenus par séquençage sont fournis en entrée du pipeline, dans des fichiers au format `.fastq` ou compréssés `.fast.gz`. Ceux-ci peuvent-être des reads pairés (paired-ends) ou simples (single-end). Dans le cas des reads pairés, le fragment d'ADN séquencé est supposé être entièrement couvert par les reads, tels que ceux-ci se superposent sur une partie de leur longueur. Le pipeline peut prendre en argument un dossier contenant tous les fichiers de reads à traiter, ou un fichier unique d'amplicons.
- **Cleaning & Trimming.** Les reads, en sortie de séquenceur, peuvent contenir un certain nombre d'erreur. Les bases séquencées sont associés, dans le fichier `.fastq`, avec un indice de qualité de séquençage. Les extrémités des reads sont souvent de qualité inférieure au reste. La première étape consiste donc à couper (trim) ces extrémités en fonction de leur qualité. Elle est effectuée grâce au logiciel AlienTrimmer [7]. Une valeur seuil de qualité et une longueur minimale de reads est définie pour le trimming.
- **Merging.** Si les reads en entrée sont des reads pairés, cette étape consiste à fusionner les reads de chaque paire. Elle est réalisée avec le programme Flash [8], auquel il faut passer en argument les fichiers de reads forward et reverse (les deux extrémités), la longueur de superposition minimale (par défaut 10 nucléotides) et la longueur de superposition maximale (par défaut 200 nucléotides). Le programme renvoie un fichier `.fastq` contenant les reads fusionnés.
- **Transformation au format fasta.** Les fichiers de reads aux format fastq sont simplement transformés au format fasta, et tous les fichiers de l'analyse sont concaténés en un seul.
- **Dereplication.** La déréplication (ou déduplication) consiste à supprimer les doublons de reads. Une seule copie de tous les reads identiques est gardée, à laquelle on ajoute l'abondance (le nombre de reads identiques) au header du fasta. Cela permet de réduire la taille du fichier et de faciliter les analyses en aval. La déréplication est effectuée avec Vsearch [4].

- **Singleton removal.** Les reads d'abondance inférieure à un certain seuil (par défaut 4) sont exclus, car ils sont considérés comme non informatifs. Ils peuvent provenir d'erreurs de séquençage, ou juste provenir de variants très faiblement représentés. Cette étape est également effectuée avec Vsearch.
- **Chimera filtering**. Certains reads peuvent résulter de la fusion de plusieurs fragments et ils sont appelés reads chimériques. Il convient de filtrer ces reads qui sont des artefacts non informatifs. Cette étape est effectuée avec Vsearch. Par défaut, le filtrage est réalisé *de novo*, mais il est possible de le réaliser à l'aide de la base de données ChimeraSlayer en option.
- **Clustering**. Les reads sont clusterisés en OTU (Operational Taxonomic Unit). Tous les reads avec une similarité supérieure à 97% sont clusterisés ensemble (Fig. 4). Chaque OTU est censé représenter les reads provenant d'une même espèce. Les 3% de différence de séquences au sein du même OTU sont considérés comme de la variabilité de séquence intra-espèce [6]. Le clustering peut être effectué avec Vsearch (par défaut) ou Swarm [9].

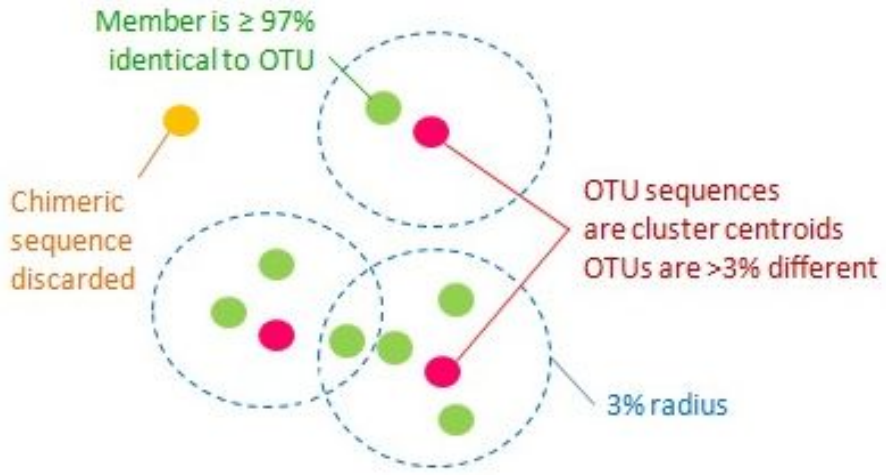


Figure 4 : *Représentation du clustering des OTU. Une séquence centrales dans le cluster est conservée comme séquence représentatrice. Toutes les séquences qui ont moins de 3% de différences sont assignées à ce cluster.*

- **Mapping.** Les reads sont alignés sur les OTUs, de manière à obtenir une matrice de comptage avec en colonnes les différents échantillons de départ (les fichiers `.fastq`), en lignes les OTUs, et en valeurs de la matrice le nombre de reads. Cette étape est réalisée avec Vsearch. La matrice est exportée au format Biom pour des analyses ultérieures.

- **Annotation.** Les OTU sont alignés sur des bases de données de séquences afin de retrouver les espèces présentes dans les échantillons. Les petites sous-unités 16S et 18S des ribosomes sont alignées sur les bases de données RDP [13] et GreenGenes [14]. Les grandes sous-unités 23S et 28S sont mappés sur les bases de données RDP et Silva [15]. Les ITS sont mappés sur les bases de données RDP, Unite [16], Findley et Underhill. Ce mapping peut-être effectué avec Vsearch (par défaut) ou avec Blastn [10]. Enfin, les résultats sont ajoutés aux matrices de comptage de l'étape de mapping dans le fichier Biom. Ce fichier Biom peut être utilisé comme fichier d'entrée par SHAMAN pour analyser les résultats.
- **MSA & Phylogeny.** Un alignement multiple est réalisé entre tous les OTUs avec Mafft [12], et une phylogénie est générée avec BMGE [11]. Cette étape n'a cependant pas pu être implémentée dans le pipeline à cause de problèmes d'exécution.

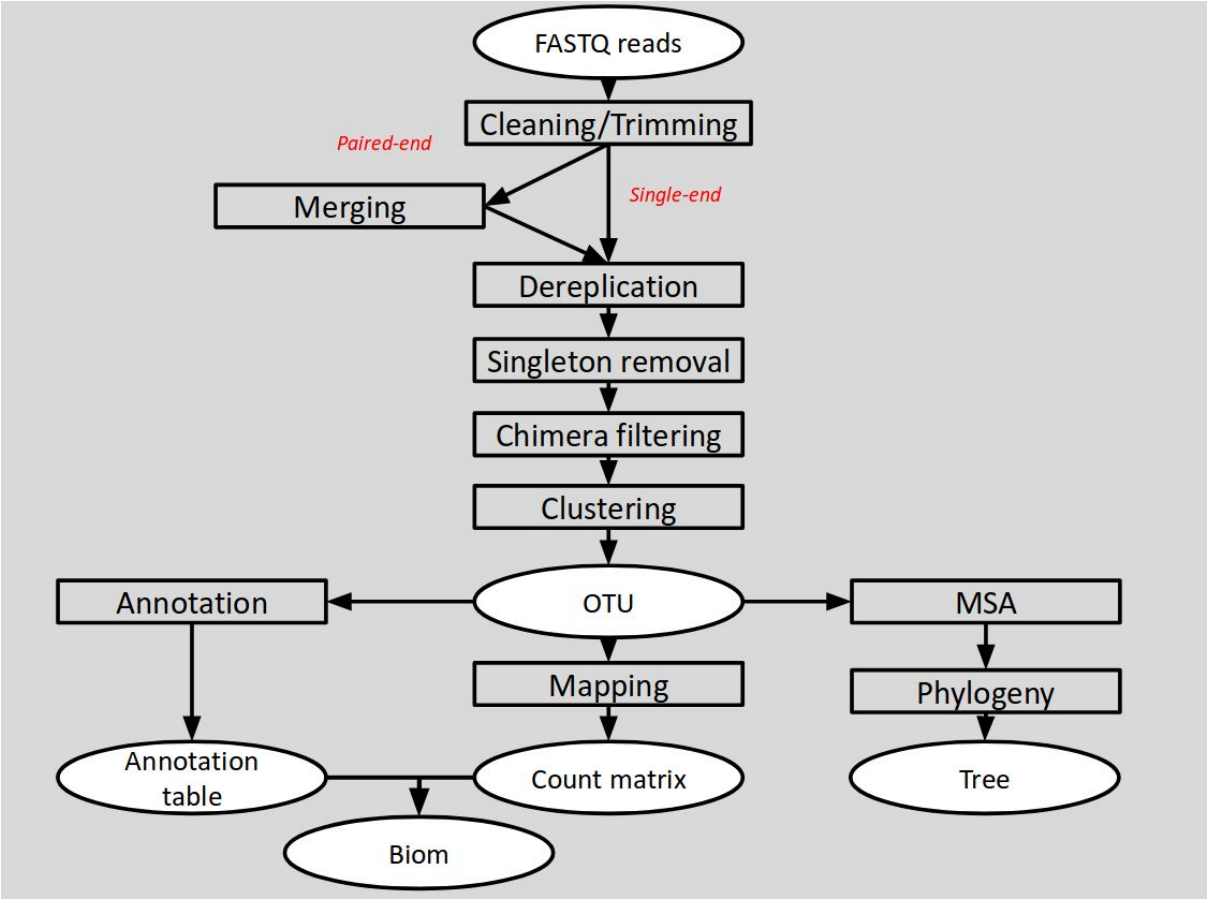


Figure 4 : *Schéma du pipeline (Source : Amine Ghozlane). L'enchaînement des tâches est le même que pour MASQUE. Le pipeline prends en entrée des fichiers de reads au format Fastq. Les reads sont filtrés et clusterisés en OTU. Le pipeline renvoie la matrice de*

comptage des reads sur les OTU, l'annotation des espèces associée, ainsi qu'une phylogénie de ces espèces (l'étape de phylogénie n'est cependant pas implémentée).

Caractéristiques du pipeline

Le pipeline implémenté en Nextflow reprend la majorité des fonctionnalités annexes de MASQUE, implémenté en bash. Une page d'aide est renvoyée avec l'option `--help` ou si aucun argument n'est entré dans la ligne de commande. Cette page d'aide décrit l'utilisation du programme ainsi que les arguments obligatoires et optionnels. Les arguments passés en ligne de commande sont également vérifiés et des messages d'erreurs s'affichent dans les cas ambigus. Enfin, si la ligne de commande est valide, le programme se lance, et l'ensemble des paramètres, ceux entrés par l'utilisateur et ceux par défaut sont affichés (Fig. 5).

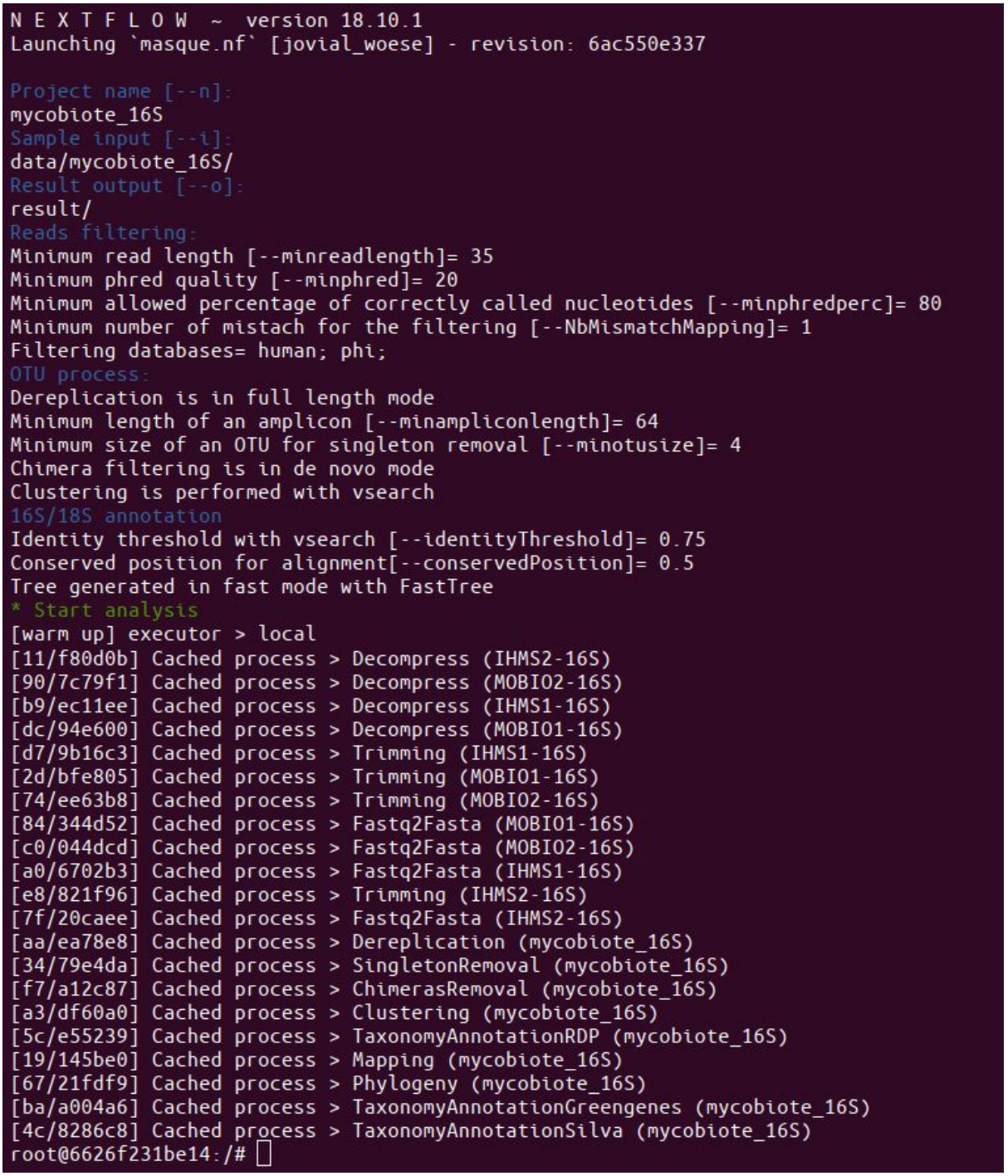


Figure 5 : *Example of display when running the pipeline.*

Pendant l'exécution, le pipeline produit des fichiers intermédiaires à chaque tâche. Tous ces fichiers sont enregistrés dans le dossier de résultats. Le pipeline est également conçu pour enregistrer certains fichiers logs (rapports d'exécution) et les fichiers logs d'erreurs (rapports d'erreurs). Ces fichiers sont aussi automatiquement enregistré dans le dossier de résultat.

Découlant directement des fonctionnalités de Nextflow, le workflow affiche les tâches en cours d'exécution, mais pas les sorties standard des programmes utilisés. Ce comportement peut être modifié dans le fichier de configuration utilisateur. Nextflow produit également des rapport d'exécution automatique, et affiche un message d'erreur si un process s'arrête. Les messages d'erreur de Nextflow sont bien détaillés et facilitent le débuggage. Enfin, l'option `-resume` de Nextflow permet de reprendre une exécution arrêtée à son dernier checkpoint.

Grâce à Docker, le pipeline peut être mis en oeuvre très rapidement. L'image Docker du workflow est hébergée sur DockerHub sous le nom *etjean/shaman_nextflow*. Seule l'installation des bases de données est nécessaire. Le téléchargement de l'image Docker et l'ouverture d'un nouveau conteneur sont automatique lors du lancement du pipeline. Cependant, Docker permet aussi un mode interactif, avec le lancement d'un terminal propre au conteneur, qui se révèle souvent plus pratique. Enfin, le répertoire de travail de Nextflow et d'autres fichiers qui auraient accidentellement pu être créés au cours de l'exécution seront supprimés avec le conteneur, ce qui évite de polluer l'environnement de l'utilisateur.

Le dépôt Github du pipeline décrit l'utilisation du workflow et son lancement avec Docker. Des données test de séquençage single-end d'ARN 16S et d'ITS, et paired-end d'ARN 16S et 18S sont fournies pour tester le programme.

Conclusion

Bien que pipeline MASQUE codé en bash implémente déjà plusieurs fonctionnalités qui sont incluses par défaut par Nextflow (parallélisation, gestion d'erreurs, génération de logs), la version en Nextflow de ce projet apporte d'autres avantages. Le langage étant spécifiquement conçu pour le développement de workflow, la maintenance, les mises à jours et le développement futur en sont facilités. MASQUE supporte actuellement la mise en oeuvre sur des clusters de calcul SGE et Slurm. Nextflow le permet par conception sans modification, ainsi que le déploiement sur des exécuteurs en *cloud computing*. Enfin, Nextflow permet la séparation entre utilisateurs et développeur en autorisant l'utilisateur à décrire des paramètres spéciaux dans un fichier de configuration sans avoir besoin de modifier le code. D'autre part, MASQUE était déjà conçu pour fonctionner dans un système de conteneur Docker. Une nouvelle image Docker a été réalisée à partir de l'ancienne pour support le fonctionnement du workflow de ce projet.

Le workflow développé au cours de ce projet reste cependant incomplet, et de nombreuses améliorations pourraient encore être apportées : La partie de génération de la phylogénie notamment comporte un problème qu'il n'a pas été possible de régler. Enfin, en vue d'une intégration dans SHAMAN, il faudrait notamment effectuer de multiple tests afin de vérifier et peaufiner la robustesse du programme, et développer son utilisation sur le cluster de calcul de l'Institut Pasteur.

Bibliographie

1. Siegwald L, Touzet H, Lemoine Y, Hot D, Audebert C, Caboche S. Assessment of Common and Emerging Bioinformatics Pipelines for Targeted Metagenomics. PLoS One. 2017;12: e0169563.

2. Visconti A, Martin TC, Falchi M. YAMP: a containerized workflow enabling reproducibility in metagenomics research. Gigascience. 2018;7. doi:10.1093/gigascience/giy072

3. Quereda JJ, Dussurget O, Nahori M-A, Ghozlane A, Volant S, Dillies M-A, et al. Bacteriocin from epidemic Listeria strains alters the host intestinal microbiota to favor infection. Proc Natl Acad Sci U S A. 2016;113: 5706–5711.

4. Rognes T, Flouri T, Nichols B, Quince C, Mahé F. VSEARCH: a versatile open source tool for metagenomics. PeerJ. 2016;4: e2584.

5. Handelsman J. Metagenomics: Application of Genomics to Uncultured Microorganisms. Microbiol Mol Biol Rev. 2005;69: 195–195.

6. Yarza P, Yilmaz P, Pruesse E, Glöckner FO, Ludwig W, Schleifer K-H, et al. Uniting the classification of cultured and uncultured bacteria and archaea using 16S rRNA gene sequences. Nat Rev Microbiol. 2014;12: 635–645.

7. Criscuolo A, Brisse S. AlienTrimmer: a tool to quickly and accurately trim off multiple short contaminant sequences from high-throughput sequencing reads. Genomics. 2013;102: 500–506.

8. Magoč T, Salzberg SL. FLASH: fast length adjustment of short reads to improve genome assemblies. Bioinformatics. 2011;27: 2957–2963.

9. Mahé F, Rognes T, Quince C, de Vargas C, Dunthorn M. Swarm v2: highly-scalable and high-resolution amplicon clustering. PeerJ. 2015;3: e1420.

10. McGinnis S, Madden TL. BLAST: at the core of a powerful and diverse set of sequence analysis tools. Nucleic Acids Res. 2004;32: W20–5.

11. Criscuolo A, Gribaldo S. BMGE (Block Mapping and Gathering with Entropy): a new software for selection of phylogenetic informative regions from multiple sequence alignments. BMC Evol Biol. 2010;10: 210.

12. Katoh K, Standley DM. MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Mol Biol Evol. 2013;30: 772–780.

13. Cole JR, Wang Q, Cardenas E, Fish J, Chai B, Farris RJ, et al. The Ribosomal Database Project: improved alignments and new tools for rRNA analysis. Nucleic Acids Res. 2009;37: D141–5.

14. DeSantis TZ, Hugenholtz P, Larsen N, Rojas M, Brodie EL, Keller K, et al. Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB.

Appl Environ Microbiol. 2006;72: 5069–5072.

15. Quast C, Pruesse E, Yilmaz P, Gerken J, Schweer T, Yarza P, et al. The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. Nucleic Acids Res. 2013;41: D590–6.

16. Kõljalg U, Larsson K-H, Abarenkov K, Nilsson RH, Alexander IJ, Eberhardt U, et al. UNITE: a database providing web-based methods for the molecular identification of ectomycorrhizal fungi. New Phytol. 2005;166: 1063–1068.