Technical Report

ELIXIR Compute Platform Technical Demonstrator

Rare Disease NGS pipeline implementation in CWL and file transfers using pure-ftpd and ELIXIR AAI -enabled sshd.

Last updated: 31.07.2019
Author: Jarno Laitinen (jarno.laitinen@csc.fi, ELIXIR FI) et al.

1 Introduction

This document describes the demonstrator implementing the variant calling pipeline on NGS data used by researchers working on rare diseases. The workflow requires paired-end sequenced raw genomic data (in FASTQ format) and reference genome data. It runs a mapping and variant calling pipeline and in turn produces unannotated gvcf files, which can be further submitted to the RD-Connect GPAP portal or analyse on their own.

The computational part of the demonstrator is implemented on the standards (GA4GH) compliant, interoperable container orchestration platform. The workflow is written in Common Workflow Language (CWL). The applications are available as Docker Containers.

People involved in this demonstrator

The following people made the main contribution to this demonstrator:

WP8 Rare Disease community:

- Raul Tonda: a data analyst, in charge of RD-Connect analysis pipeline development.
- Leslie Matalonga: a Clinical Genomics Specialist responsible forcoordinating differen t activities related toELIXIR RD Community, EXCELERATE WP8 activities and RD-Connect GPAP.

Compute Platform:

- Jarno Laitinen forexecuting CWL files and the client-side scripting based on existing code.
- Ania Niewielska (EBI, UK) and Alex Kanitz (Unibas) for supporting TES and WES.

Used data

For the demonstration of RD pipeline, the following publicly available DNA sequence(FASTQ pair-end) data, which can be downloaded without restrictions:
`U5c_CCGTCC_L001_R1_001.fastq.gz`
`U5c_CCGTCC_L001_R2_001.fastq.gz`

The raw data files from real world applications can be in the order of tens of terabytes and therefore data transfer step can be time consuming. Therefore for local development it is advisable to keep a local copy of the files.

Reference genome: `hs37d5.fa.gz`
the known indel and sites files:`Mills_and_1000G_gold_standard.indels.b37.vcf and`
`dbsnp_138.b37.vcf.gz`

Currently, the use case data for production level usage come often via Aspera server (commercial software), which can be installed using a container without licensing costs. Later the aim is to get genomes from EGA.

Once data are processed, the resulting gvcf files will be returned to RD-Connect for further annotation and can be included in the RD-Connect GPAP portal

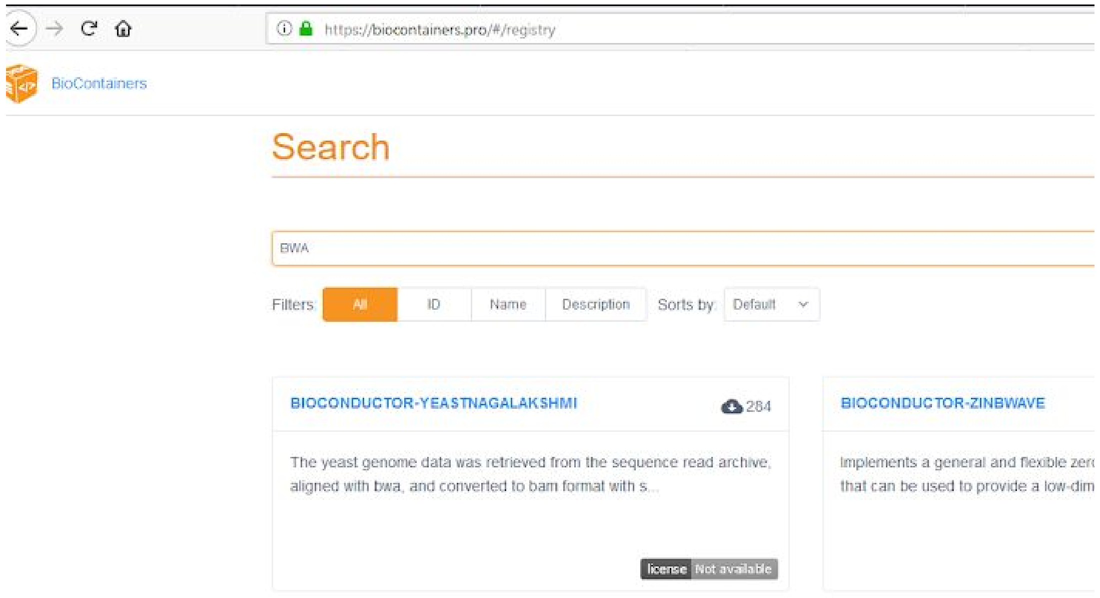
File transfer implementation is described below. However, there is no one or a standard way to transfer the data.

2 The Components

Containers

Currently the most popular method for using container functionality in Linux is to use dockerised containers. For more complete documentation about Docker, see the official Docker documentation.

The execution platform requires that the containers are on a public repository that a CWL script can fetch the container (docker pull). Many applications are already in the containers and many of those can be searched with bio.tools registry. However, it is possible to use own containers.by providing a docker file and docker environment. The resulting image built from the dockerfile can be pushed to Docker Hub.


https://biocontainers.pro/#/registry
BioContainers
Search
BWA
Filters: All ID Name Description Sorts by: Default
BIOCONDUCTOR-YEASTNAGALAKSHMI
284
license Not available
BIOCONDUCTOR-ZINBWAVE

The yeast genome data was retrieved from the sequence read archive, aligned with bwa, and converted to bam format with s...

Implements a general and flexible zer
that can be used to provide a low-dim

Figure 1: Biocontainer’s registry search web interface.

Biocontainers.pro is a search engine for existing containers. Metadata and visibility of the content of a container is limited though. The user has to trust container and therefore for sensitive data processing some extra cautions are needed. In addition to malicious modifications the software on the container might have issues e.g. regarding updated. Other option is to install software to a container from scratch. The container must be saved in a public repository that the workflow execution system can download it.

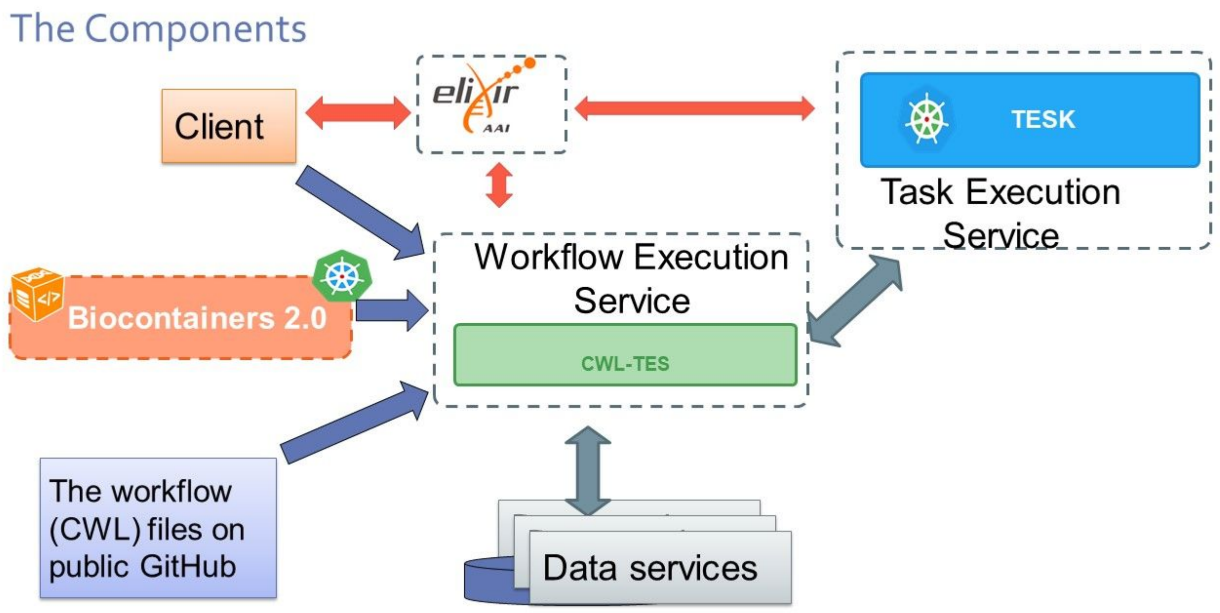


Figure 2: The system used in this Demonstrator consisting of the client side, ELIXIR AAI, Docker repositories, CWL files, data transfer services and GA4GH compliant Workflow and Task Execution Services.

The WES and TES components are implementations of GA4GH Cloud and AAI workstream standards[1]. The development work is on-going and one of the main developments concerns storage service behinds the scenes since this reference implementation used FTP server. Local file server implementation already exists at EBI.

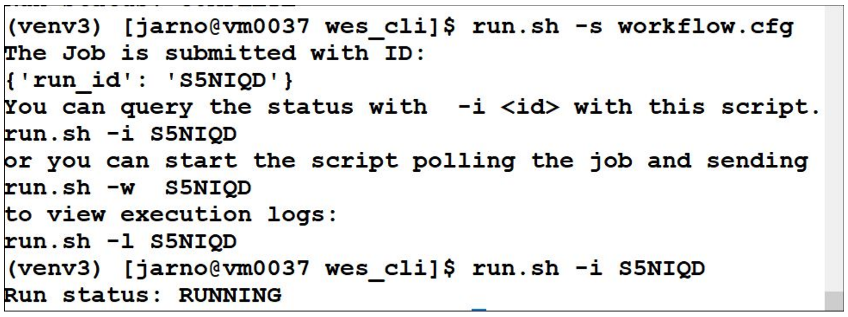
Client

For this demonstrator a basic client implementation was further developed[2]. The client interacts with the Workflow Execution Service API. The customer can use the client instead of Swagger WEB user interface, where writing the parameters is not that nice and can be lost when refreshing the web page.

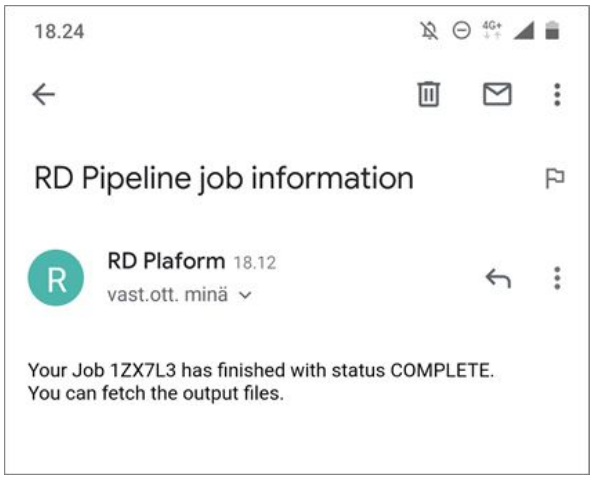
The existing very basic client implementation was complimented with AAI token fetching functionality, advanced configuration files and query options.

[1] https://elixir-europe.github.io/cloud/
[2] https://github.com/jarnolaitinen/WES_client

The client has configuration files to define input parameters for the workflow as well as the workflow CWL main file Only the main workflow file needs to be given. User's e-mail address can also be configured to inform when job is ready.



```
(venv3) [jarno@vm0037 wes_cli]$ run.sh -s workflow.cfg
The Job is submitted with ID:
{'run_id': 'S5NIQD'}
You can query the status with  -i <id> with this script.
run.sh -i S5NIQD
or you can start the script polling the job and sending
run.sh -w  S5NIQD
to view execution logs:
run.sh -l S5NIQD
(venv3) [jarno@vm0037 wes_cli]$ run.sh -i S5NIQD
Run status: RUNNING
```



18.24

RD Pipeline job information

RD Plaform 18.12
vast.ott. minä

Your Job 1ZX7L3 has finished with status COMPLETE.
You can fetch the output files.

Figure 3: A command line output for the python client and email from its job monitoring process.

As a job may run for several hours, it is nice to get an alert when a job is finished. This is implemented into the client tool as a loop. Thus, user can leave the client tool running for example on a screen session without ssh connection being terminated. However, the token has to be valid and is created if needed again as there is no automatic token renewal mechanism.

Here is an example video.

4 The workflow

Most of the pipeline is done in a certain sequence, though there are some steps, which could be done concurrently.

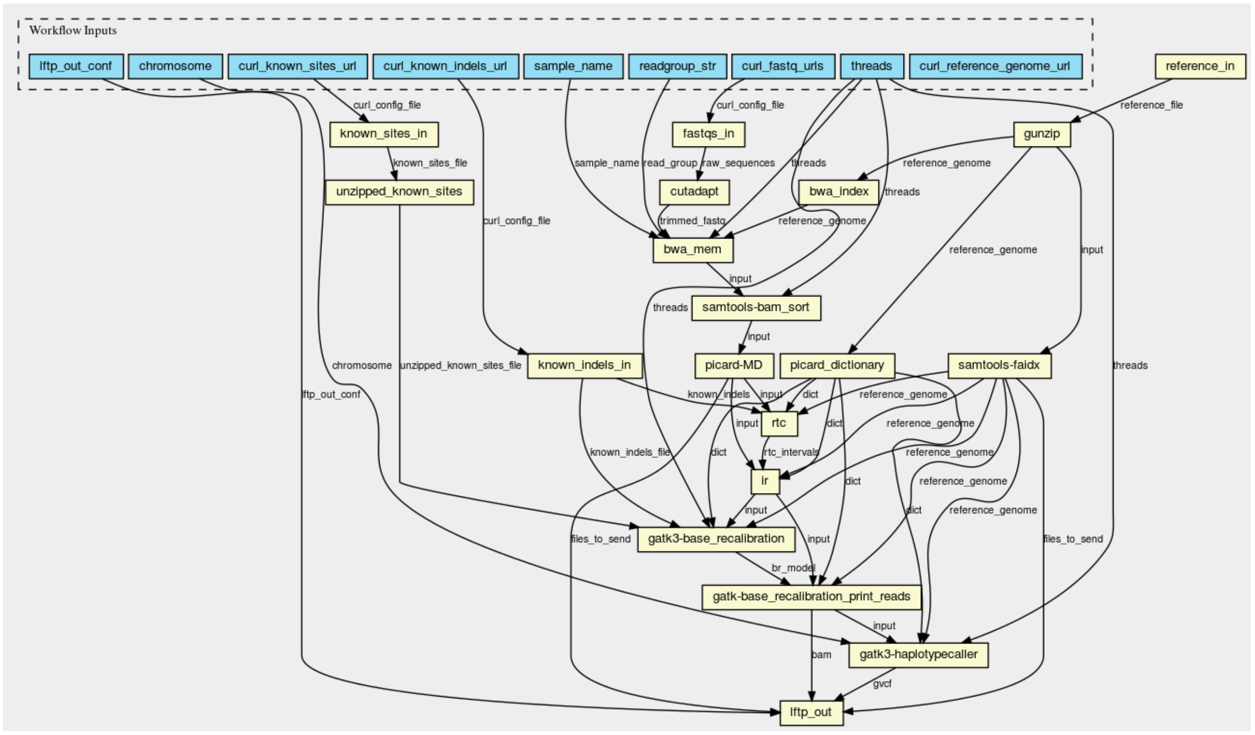


Figure 4: CWL workflow graph produced with https://view.commonwl.org/

The files are in the repository[3], which also has the files for NextFlow implementation of the same pipeline.

Briefly, Cutadapt tool was used to remove sequencing adaptors from the FASTQ files. It can be run in paired mode (-p) so that there is only one output file (trimmed FASTQ). BWA index and samtools faidx makes the index files of the reference genome, which was unzipped with gunzip step [4]. Picard sequence dictionary makes .dict file.This kind of steps could be done earlier once as it does not depend on the input files. BWA-mem maps the trimmed reads to human reference genome. Samtools sort sorts the BAM file.

Picard with markduplicates marks and removes duplicates. It produces also output metrics file. Base quality recalibration was done with Genome Analysis toolkit (GATK v.3 https://software.broadinstitute.org/gatk/). The containers were fetched from Broad Institute's public repository. GATK has four different steps before producing the unannotated gvcf file. In addition to the variant output files, which are transferred in the final step, it is useful to have the indexed I files since some programs use them for downstream analyses (such as annotation).. It is also useful to keep the alignment files to visually inspect some variants.

[3] https://github.com/inab/Wetlab2Variations/tree/master/cwl-workflows/demonstrator
[4]https://gatkforums.broadinstitute.org/gatk/discussion/2798/howto-prepare-a-reference-for-use-with-bwa-and-gatk

CWL scripting language

Common Workflow Language (CWL)[5] is a one of many workflow languages aiming to be portable across the systems. There are other reference implementations such as WDL to run the scripts. CWL-TES implementation uses CWL underneath.

For many bioscience applications, CWL scripts already exist in public repositories to start with. Additionally, There are some useful tutorials for learning[6].

The CWL file specifies resource requirements such as number of cores, RAM size and directory capacity.The requirement can be put as requirement or hint. It depends on the implementation how the hints are handled. If the requirements are too large the job cannot be executed. If there requirements are too small, then the application can fail in case it does not get enough resources. Those in turn may be correlated data size. Additionally resources for JavaScript or Docker containers need to be indicated in the requirements.

5 Execution Platform

The implementation of Global Alliance for Genomics and Health (GA4GH) compatible cloud and AAI system was created in 2018. It consists of the Workflow Execution Service (WES) and Task Execution Service (TES). This implementation which was used in the Demonstrator is still in proof-of-concept status.

The development team works in ELIXIR Implementation Study and EOSC-Life funded tasks and they have done architecture roadmap documents. There are roadmaps for further development so this document does not aim to list missing features or bugs. The collaboration continues within ELIXIR Programme with Implementation Study funding as well as in EOSC-Life (Task 7.4.2 GA4GH-Compatible Workflow Platform). One of the key issues is the storage environment. Also currently WES neither allows user to specify the TES norfaciliate intelligent load balancing. WES also does not do other attribute based direction. The idea is that there are several TES systems running, where one or more WES system can forward the job. One on-going (summer 2019) project is focusing on the task distribution login to different TES instances.

[5] CWL Description https://www.commonwl.org/v1.0/CommandLineTool.html
[6] CWL Tutorials: https://gitlab.ub.uni-bielefeld.de/c/eccb18tutorial/tree/master/workflow
https://www.melbournebioinformatics.org.au/tutorials/tutorials/cwl/cwl/
http://www.commonwl.org/user_guide/

File Transfer

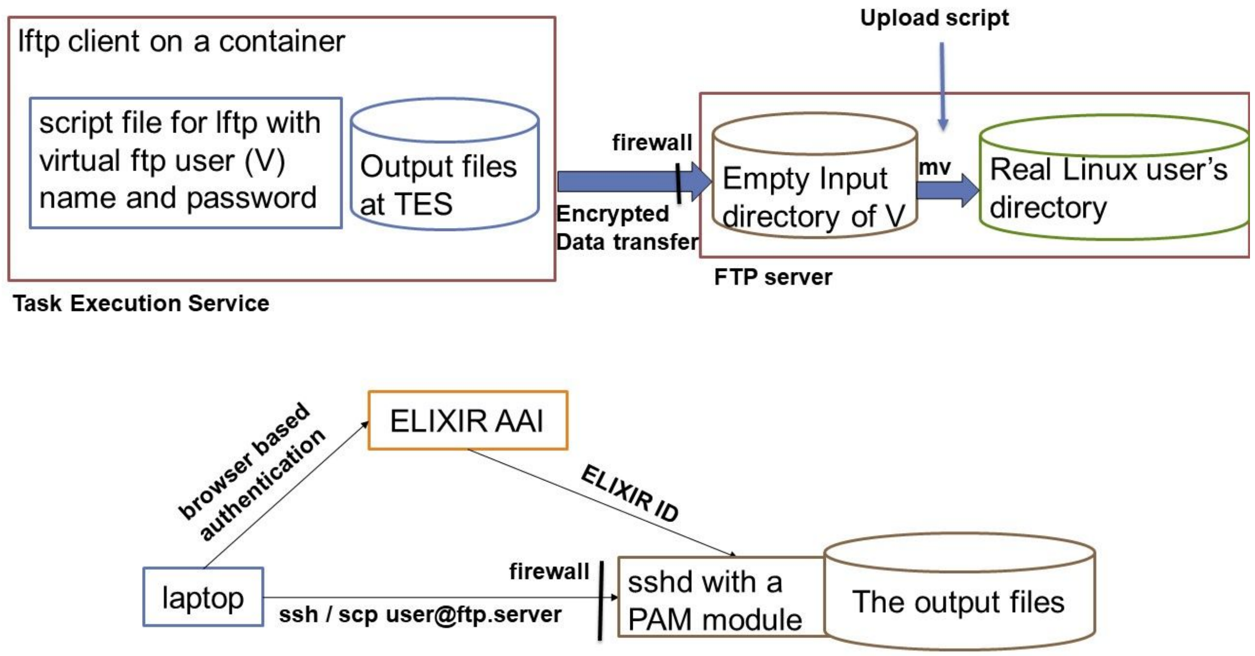

lftp client on a container
script file for lftp with virtual ftp user (V) name and password
Output files at TES
Task Execution Service
firewall
Encrypted Data transfer
Upload script
Empty Input directory of V
mv
Real Linux user's directory
FTP server
ELIXIR AAI
browser based authentication
ELIXIR ID
laptop
firewall
ssh / scp user@ftp.server
sshd with a PAM module
The output files

Figure 5: Upper part: Output files transfer from Task Execution service to pureftp server, where upload script moves the files from the virtual user to normal user account. Lower part: Output file transfer from researcher's laptop with scp using ELIXIR ID and OpenID based PAM module.

The file transfer mechanisms used here for the input and output files are not compliant withGA4GH standards, but just standard file transfer tools in CWL jobs for this demonstrator purpose.

The public reference genome files are on ftp servers and thus can be accessed with various tools. Here was used curl, but could have been for example lftp client, which was used for the input file transfer. Both commands take a script file as input file. That in turn was located on a web server, which the user could administrate. Typically all options can be also given as command line parameters for the CWL workflow job. Then the amount of inputs in turn increase.

The output files are transfered to pure-ftpd FTP server, which was installed on a virtual machine for this demonstrator purpose. Proftpd is other quite similar ftp server. The FTP implementation did support SSL encryption for the data transfer and allows to configure server certificates. Dealing with the certificates is not very simple as the client side needs to support host certificate authority. Here was used the username and password based authentication, but it was for a virtual FTP user. That account does have only access to the FTP server, not to ssh, and there is basically an empty directory for writing files. Then an automatic script, running under root permissions, moves the files from the directory to another directory, which is not accessible by the virtual user. The script is called by the FTP server. In the demonstrator file was copied to a real user account's home directory. Then the researcher can login and fetch the files. It could be FTP as well, but here was used scp, which encrypts data over ssh. Authorisation file was based on user's ELIXIR ID but it could be of course be based on the normal ssh keys or password. The server mapped the ELIXIR ID to a real linux user account. The scp used a PAM module, which is introduced in the following AAI section. Thus, no password or user account has to be communicated to the user. The Linux user account could be automatically created for all members of a certain Perun group, but that was not implemented in this demonstrator.

As there are not many addresses which need access to FTP server, those are feasible to put into firewall. TES and WES admins can give the IPs for the FTP server admin. Also for the ssh/scp interface a firewall is possible to configure for individual IPs or IP ranges from where the researcher is going to download the files. If the FTP server is running on a virtual machine, the cloud middleware might have firewall settings as well and that might be closed by default.

AAI

This Demonstrator also showed usage of Python PAM module done at ELIXIR CZ [7]
The python client script contains AAI service endpoint specific secret key, which interacts with ELIXIR AAI. The client authentication process produces a link and QR code, which either of them can be used to open browser to authenticate in ELIXIR AAI service, which support various authentication methods. Typically user uses home organisation's username and password to authenticate.

The client software stores the OpenIDtoken into the user's directory. The token lifetime was set to six hours in this test environment. In production usage it will take longer time for large genome files no matter how powerful the computing servers are. Therefore either the token lifetime has to be long or there must be a renewal mechanism, which should not need user interaction.

This test environment uses a single Perun group to authorise usage on TES. The group manager can add to the group. The group can be used as resource usage allocation purpose as well i.e. a task execution site would authorise that group to use the computing environment. In the production usage Rare Disease Community or organisation could negotiate permission to use certain execution sites, which would authorise the Community members to use the resources.

WES does not allow submit a job without a valid token. A person who has not submitted the job get 403 error if trying to access the job information.

[7] https://github.com/ICS-MU/pam_oauth2_device

Appendix 1: Technical configurations

The Client side Execution environment

In this Demonstrator CWL files were developed and tested on a CentOS 7 virtual machine running on OpenStack cloud. There was installed python virtual environment, which allows to select python version and install tools with pip command. Then user can setup own environment.

```
$  source /home/jarno/venv/bin/activate
(venv) [jarno@jlcwl ~]$ python -V
Python 2.7.5
```

To install cwltool

```
pip install --user cwltool
```

And then you can run

```
cwltool workflow.cwl workflow.yml
```

Or a single CWL fine assuming that the input files of its .yml file are existing. As the .yml files specify the path you have to tune those. The output files comes to the same directory. However, by default the temporary files are written under /tmp. The directory can be changed by defining a different path. See cwltool help. You should check that those temporary files are cleaned after the job has finished or clean as you might run out of the disk space. Otherwise you have to clean them manually.

curl command takes a simple input file:

```
curl -o U5c_CCGTCC_L001_R1_001.fastq.gz
url="ftp://<insert here server and path>
/U5c_CCGTCC_L001_R1_001.small.fastq.gz"
-o U5c_CCGTCC_L001_R2_001.fastq.gz
url="ftp://<insert here server and
path>/U5c_CCGTCC_L001_R2_001.small.fastq.gz"
```

In case FTP requires username and password, those can be added like:

```
username:password@ftp_server_access
```

In the Demonstrator there are configuration files for the python command-line client. Those contain references to a web server, which serves input files for the workflow download clients. Those are files will contain the URLs for the real data files.

In case one wants to run a web server to serve the parameter files, which the workflow.cwl files uses, it can be as simple as the following python script

```
import SimpleHTTPServer
import SocketServer

PORT = 8000

Handler = SimpleHTTPServer.SimpleHTTPRequestHandler

httpd = SocketServer.TCPServer(("", PORT), Handler)

print "serving at port", PORT
httpd.serve_forever()
```

However, that is not using any authentication etc. and allows to access all files on that server. For temporary usage and with firewall open to WES and TES servers only, it might do. It can be started with screen command that it stays running if ssh connection breaks, but that is just needed when the setup downloads the input files.

Of course, you can download the input files locally and those are then referred from the workflow.yml file. The reference genome files are probably downloaded from a server without authentication, but the files to be analysed can be sensitive data files, which are downloaded with a special manner which requires own file transfer logic.

For the output file transfer was used pureftpd on the server side and lftp as the client.
The client can also take parameter file such as

```
set dns:order inet; set ftp:ssl-allow true
# set ftp:ssl-force true
# set ftp:ssl-protect-data true
# set ssl:ca-path /etc/ssh/certs/terena.pem
set ssl:verify-certificate no
open <ftp server address>
user ftp_virtual_username  password
mirror -R
bye
```

In this example, all files from the directory will be copied to the server into the virtual user's default directory. Naturally directories can be used.

The FTP server side configuration

The server side can also use a host certificate, which the organisation needs to order from its certificate authority (CA). Then the transfer can be encrypted and certificate authorized. However, that might require tuning on various files as the client side needs to know the server side CA's certificate as well.

Pureftp configuration file is under `/etc/pure-ftpd` .There you can configure the TLS setting and certificate.

In this Demonstrator was used a virtual user account, which used just for quick receiving of the files. Every file will be automatically copied to another directory, where a normal user account can access. The script, which is called by pure ftpd can be for example put into the `/etc/pure-ftpd` and be something like

```
#!/bin/bash
if [ "$UPLOAD_VUSER" == "virtual_username" ]; then
  # echo "$UPLOAD_VUSER uploaded file $1" >> /var/log/job.log
  if [ ! -d /home/$UPLOAD_USER/outputs ]; then
     echo `mkdir -p /home/$UPLOAD_USER/outputs`
     echo `chown $UPLOAD_USER:$UPLOAD_USER
/home/$UPLOAD_USER/outputs`
  fi
  echo `mv $1 /home/$UPLOAD_USER/outputs/` >> /var/log/job.log
  echo `chmod 600 /home/$UPLOAD_USER/outputs/$1` >> /var/log/job.log
fi
```

In this Demonstrator was also used the ELIXIR CZ made OpenID python implementation [8] for ssh. Therefore researcher can login with ELIXIR AAI based authentication mechanism without delivering password from the FTP server. The sshd process can be started into another port if the default is not wanted to be changed. FTP server operator has to create the user account. Then it can be used with normal Linux scp client :

```
scp -P port user_account@server_ip:/path/file .
```

[8] https://github.com/ondrejvelisek/pam_oauth2_device (README contains configuration information)

A new client has to be registered to ELIXIR AAI. First it can get a test status, which is 30 days period, which can be easily expanded. aai-contact@elixir-europe.org supports when filling the cryptical form [9] and defining the necessary attributes. For the ssh client is enough to know user's ELIXIR ID (ad scope : openid), which is mapped to a local Linux account user name in a configuration file on the FTP server. That file could be automatically generated from a Perun group information, but that was not implemented in this Demonstrator.

[9] https://login.elixir-czech.org/oidc/manage/dev/dynreg/new