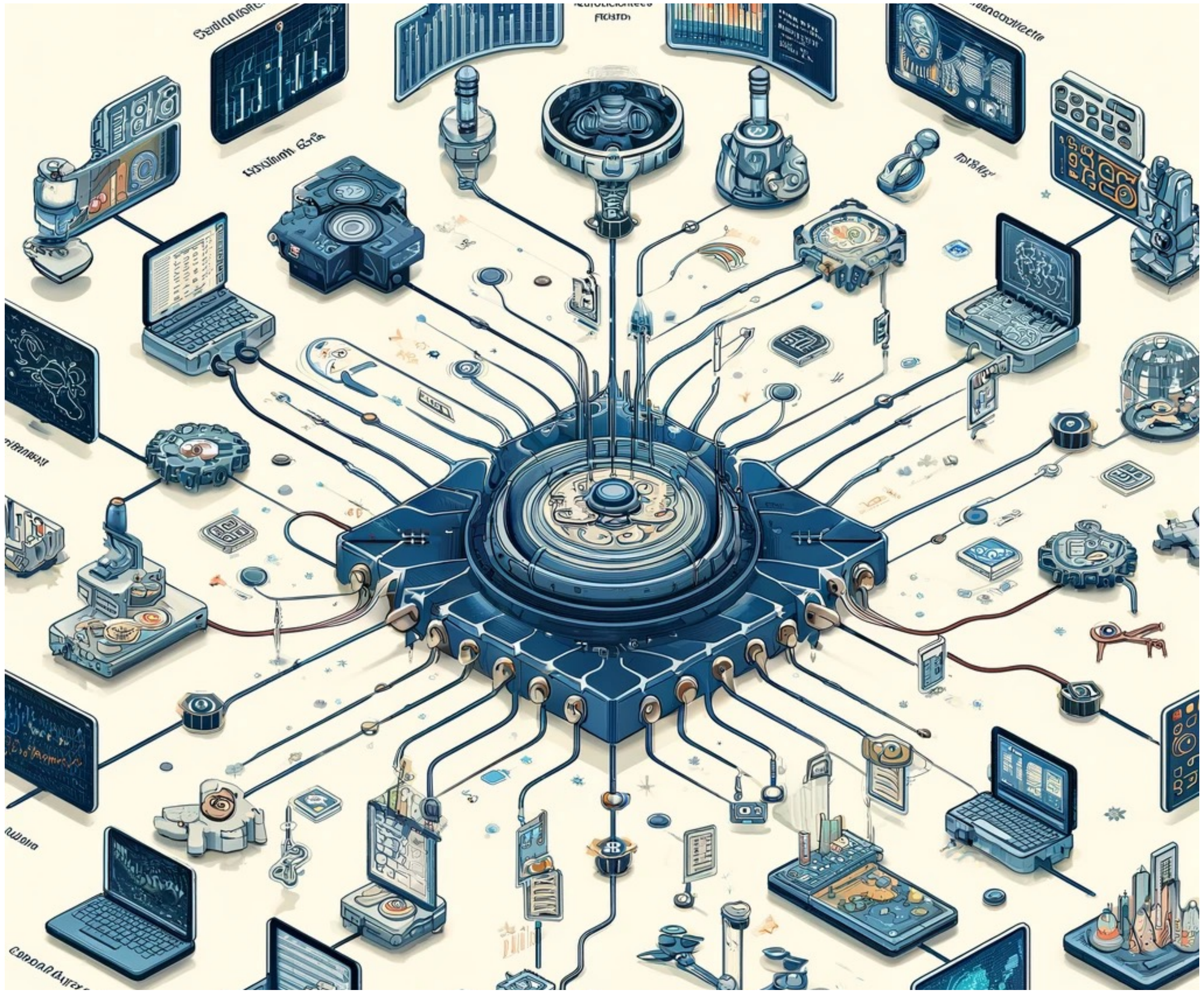
Exploring Scientific Workflows with CWL and dispel4py

Module 1.a

- Dr. Rosa Filgueira
- Lecturer at the School of Computer Science
- University of St Andrews
- rf208@st-andrews.ac.uk
- rosa.filgueira.vicente@gmail.com



Module 2.a- Creating Workflows with CWL

1. Overview
   1. What is CWL
   2. Why was created
   3. Components
   4. File structure
   5. An example of a bioinformatics use case
   6. Setup
   7. Google Colab CWL tutorial

2. Hand On- Exercises
   1. Building CWL Tools in Google Colab
   2. Creating a CWL workflow in Google Colab

1. Overview

COMMON
WORKFLOW
LANGUAGE

What is CWL ?

- It is a way to describe command line tools and connect them together to create workflows
- Scripting-Glue type
- Semantic Focus
  - With CWL every component is given a formal description in a YAML format
- We can run CWL workflows with different tools:
  - cwltool, cwl-runner, toil, arvados ….

- Why do we want to learn to use CWL?
  - Explicit IO
  - Repeatability, modularity and scalability
  - Parallelism and performace

Why was it created ?

- Stated in 2015
- Community based standard
- Collaborations (innovation)
- Publications – reproducibility

- Who is using it ?

wellcome trust sanger institute
Galaxy PROJECT
Global Alliance for Genomics & Health
HARVARD T.H. CHAN SCHOOL OF PUBLIC HEALTH
UCSC
Institut Pasteur
BROAD INSTITUTE

Components

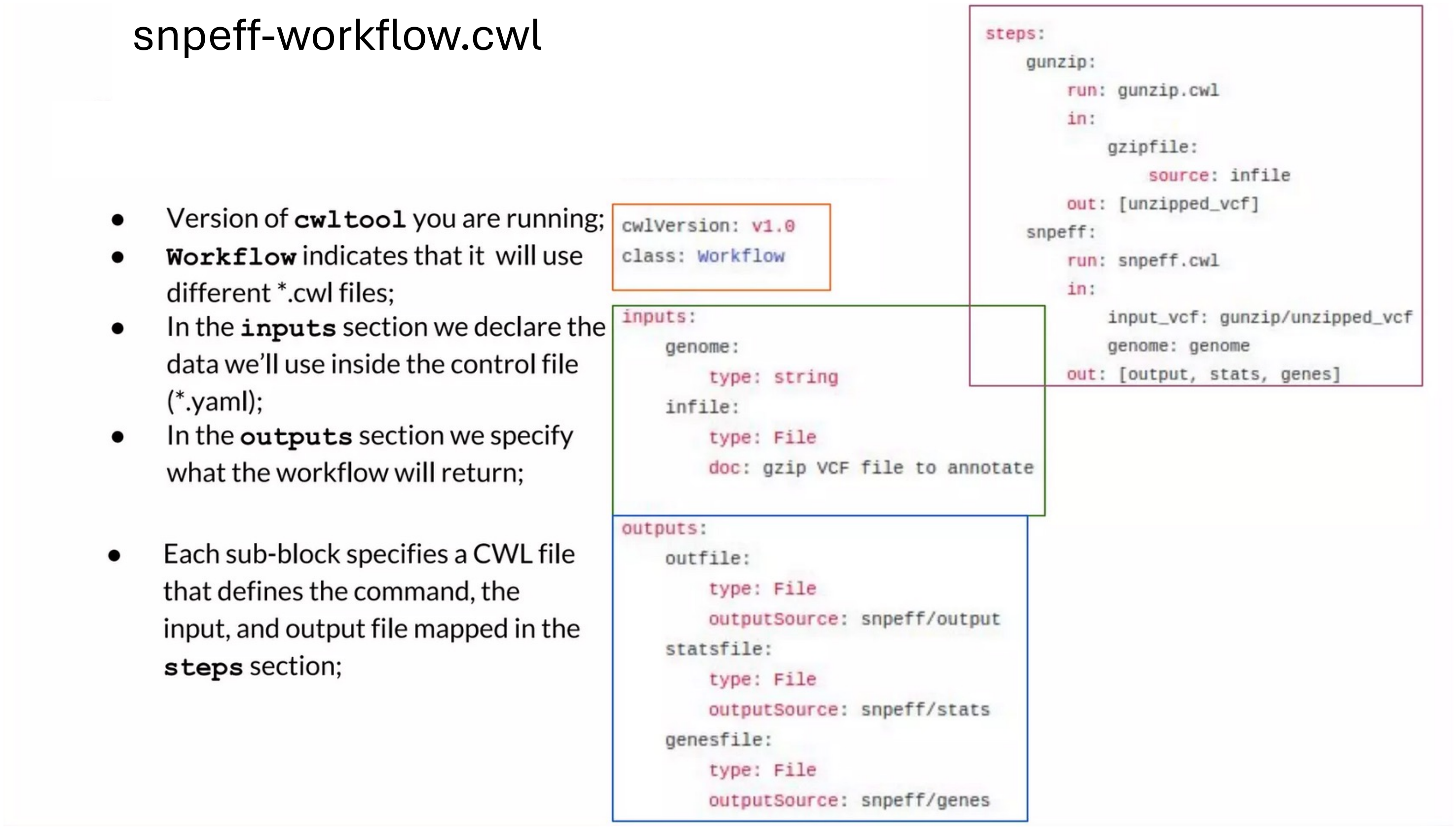
- a CWL file (.cwl)
  - Describes what is going to run and what inputs the program takes
- a YAML (.yml) file
  - Holds the values that the workflow will be executed with

- A ‘tool’ is a task to perfom within a CWL workflow
- CWL is written with YAML:
  - It is similar to JSON
    - easier to work with because it is more human readable.
  - YAML is based on key: value pairs
    - where each key is a string (text) and each value is a primitive type, an array, or an object.

File Structure

- cwlVersion:
  - describes the version of cwl being used
- class:
  - describes what the program is (e.g. CommandLineTool,Workflow)
- baseCommand:
  - provides the name of the program that will actually run
- inputs:
  - declares the inputs of the program
- outputs:
  - declares the outputs of the program
- records:
  - declares relationships between programs/parameters
- requirements:
  - declares special requirements needed by the program such as dependencies
- steps:
  - used for the actual creation of workflows and linking programs together.

An example of bioinformatics use



snpeff-workflow.cwl

- Version of **`cwltool`** you are running;
- **`Workflow`** indicates that it will use different *.cwl files;
- In the **`inputs`** section we declare the data we'll use inside the control file (*.yaml);
- In the **`outputs`** section we specify what the workflow will return;

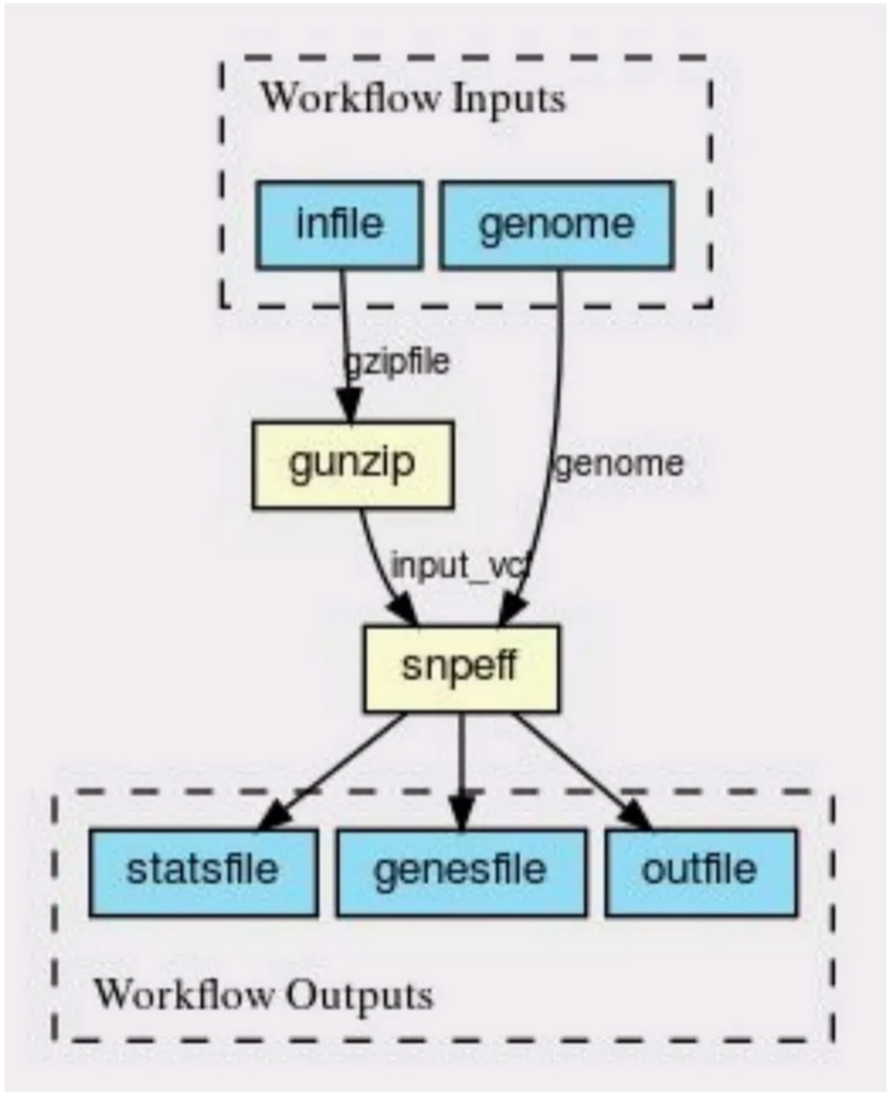
- Each sub-block specifies a CWL file that defines the command, the input, and output file mapped in the **`steps`** section;

```
cwlVersion: v1.0
class: Workflow
```

```
inputs:
    genome:
        type: string
    infile:
        type: File
        doc: gzip VCF file to annotate
```

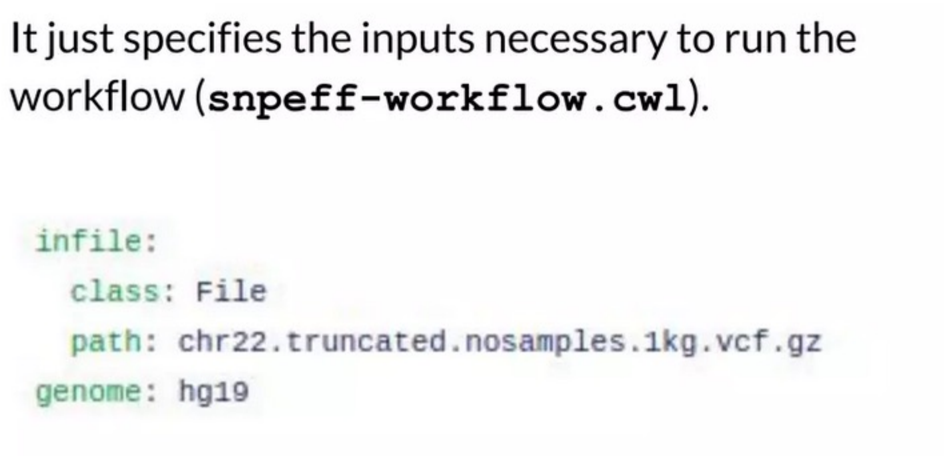
```
outputs:
    outfile:
        type: File
        outputSource: snpeff/output
    statsfile:
        type: File
        outputSource: snpeff/stats
    genesfile:
        type: File
        outputSource: snpeff/genes
```

```
steps:
    gunzip:
        run: gunzip.cwl
        in:
            gzipfile:
                source: infile
        out: [unzipped_vcf]
    snpeff:
        run: snpeff.cwl
        in:
            input_vcf: gunzip/unzipped_vcf
            genome: genome
        out: [output, stats, genes]
```



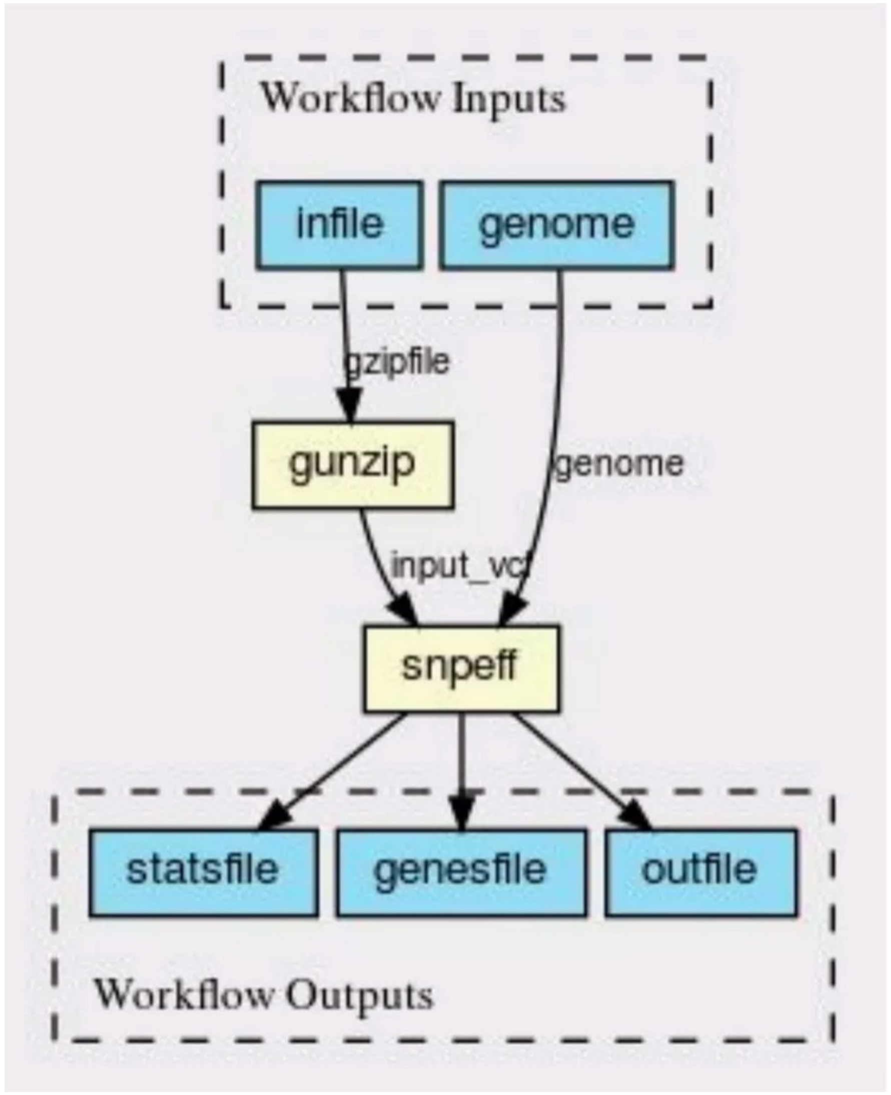
An example of bioinformatics use

1kg-job.yml

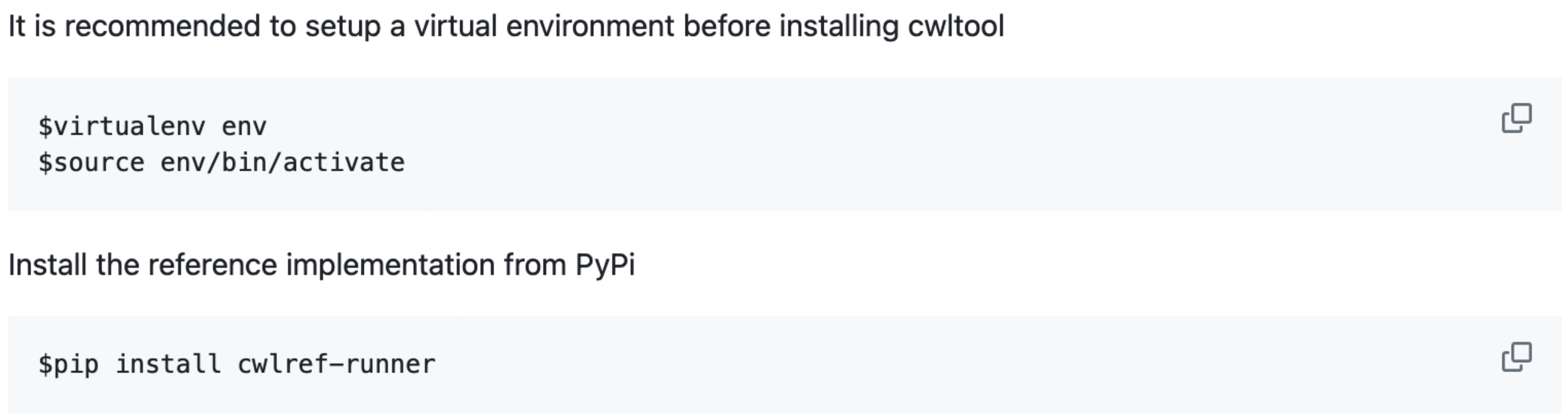


It just specifies the inputs necessary to run the workflow (**`snpeff-workflow.cwl`**).

```yaml
infile:
  class: File
  path: chr22.truncated.nosamples.1kg.vcf.gz
genome: hg19
```



Setup



It is recommended to setup a virtual environment before installing cwltool

```
$virtualenv env
$source env/bin/activate
```

Install the reference implementation from PyPi

```
$pip install cwlref-runner
```

We can also install and use instead cwltool, which another tool for running cwl workflows

```
$pip install cwltool
```

https://github.com/screx/cwl-tutorial

Lets follow our CWL tutorial

- Google Colab Notebook

2. Hands-On Exercises

COMMON WORKFLOW LANGUAGE

Exercise 1: Building CWL Tools in Google Colab

Go to your copy of Google Colab Notebook (at the end) and create CWL `tools' for wrapping :

GREP :

```
$grep "hello" helloworld.txt
hello world
```

WC :

```
user: $ wc -l somefile.txt
# prints number of lines in somefile.txt to stdout
```

TAR :

```
user $ tar -xvzf some.tar.xz
```

Exercise 1: Building CWL Tools in Google Colab

GREP Tool: Follow the instructions/code from here

1. Define inputs for the search term and the file to search in.
2. Specify the base command and arguments to reflect GREP's syntax

WC Tool: Follow the instructions/code detailed here

1. Set up parameters to count occurrences.
2. Configure the output to capture the count in a designated file

TAR Tool: Follow the instructions/code detailed here

1. Detail the input as the compressed file
2. Configure the output as the uncompressed content.

1. Note: You can use the input files that you have in ‘exercise/cl-tools/’ folder
   1. E.g. ‘exercise/cl-tools/grep/

Exercise 2: Creating a CWL workflow in Google Colab

- Design a workflow that integrates the GREP, WC, and TAR tools.
    - The workflow should sequentially uncompressed a file;
    - Search for a string ;
    - And count the occurrences, with the result saved in count.txt.
- Follow the instructions/code detailed here

Workflow Steps:

1. **Step 1:** Start with the TAR tool to uncompress the input file.
2. **Step 2:** Use the GREP tool to search for the desired string in the uncompressed data.
3. **Step 3:** Apply the WC tool on the output of GREP to count the occurrences and output to count.txt.

Workflow Execution:

1. Ensure that each tool's output is correctly piped as the input to the subsequent tool.
2. Set up the final output file count.txt to store the count from the WC tool.

Note: Remember to test each CWL tool individually before integrating them into the workflow.