Software Workflows and Infrastructures for Precision Oncology

2

Waleed Osman and Alessandro Laganà

Abstract

Precision oncology mainly relies on genetic and molecular patient profiling from high-throughput sequencing data. The necessity to process and analyze large volumes of data has led to the development of robust computational tools and methods. The most challenging aspect in the implementation of a precision oncology workflow involves proper handling of large volume of data, while ensuring the results are reproducible and replicable. In this chapter, we provide a detailed description of the various tools available for the design and implementation of a precision oncology pipeline along with the technical considerations to make to utilize these tools effectively. We then provide a guide to the development of a precision oncology pipeline, with a specific emphasis on the software workflows and infrastructure needed.

W. Osman
Department of Genetics and Genomic Sciences, Icahn School of Medicine at Mount Sinai, New York, NY, USA

A. Laganà (✉)
Department of Genetics and Genomic Sciences, Department of Oncological Sciences, Mount Sinai Icahn School of Medicine, New York, NY, USA
e-mail: alessandro.lagana@mssm.edu

Introduction

Precision oncology is an innovative research area that has introduced a novel approach to cancer care, where diagnosis, prognosis, and therapy are informed by genetic and molecular profiling of the individual patient, rather than being based on a *one-size-fits-all* approach [1–4]. This landmark paradigm shift has been enabled in recent years by the reduced cost of next-generation sequencing (NGS) technologies and a myriad of ad hoc tools and software applications developed in order to analyze the data generated [5, 6]. The explosion of tools and methods as a response to the more widely available multi-omic data sets has created a challenge in terms of reproducibility, interoperability, and standardization. Tools created for the analysis of genomic, proteomic, transcriptomic, and other omic data are typically written in one or a combination of three different styles: Command Line Interface (CLI), Application Programming Interface (API), or Graphical User Interface (GUI) [6]. Combining and ensuring reproducibility of these disparate application types has proven to be a major challenge for biologists as they often will require a deeper knowledge of software application development norms and techniques as well as greater computational capabilities. The absence of widely accepted best practices regarding software and database

© Springer Nature Switzerland AG 2022

A. Laganà (ed.), *Computational Methods for Precision Oncology*, Advances in Experimental Medicine and Biology 1361, https://doi.org/10.1007/978-3-030-91836-1_2

utilization has contributed greatly to irreproducibility, resulting in many man hours and compute cycles wasted in attempting to recreate past efforts [7].

As a remedy to this, a number of workflow management systems (WMS) and executors for running these workflow systems have been developed, such as Snakemake, Nextflow, the Workflow Description Language (WDL) (https://openwdl.org/), and The Common Workflow Language (CWL) [8–10]. Infrastructure enabling the execution of these workflows have also been developed such as Arvados (stand-alone, deployable, open-source), and Broad Institute's Terra Bio Cloud Platform (web based) [11, 12].

These infrastructure and software solutions are able to organize and process large volumes of genomics data enabling scientists to discover ever deeper insight into biological data. Today, with the use of CWL, Arvados, and Cromwell (https://github.com/broadinstitute/cromwell), and facilitated by virtual servers on cloud infrastructure, bioinformaticians and savvy data engineers can write and implement a precision medicine pipeline while maintaining reproducibility and interoperability. In this chapter, we will introduce several bioinformatics workflow management systems and the infrastructures to execute them.

Workflow Management Systems and Languages

Workflow management systems (WMS) are essential in the processing of large sets of patient's genomic data. WMS are tools developed to facilitate the orchestration and execution of computational processes in an optimal and efficient manner. In bioinformatics, these systems integrate various discrete command-line tools into one workflow for the rapid development of pipelines, which can be deployed across a variety of infrastructures and environments. Utilizing a WMS ensures ease of set-up and the ability to monitor performance of individual predefined tasks. These workflows are often linear but can also be dynamic or run in parallel. Table 2.1 provides a list with the most used WMS along with their URLs.

CWL: Common Workflow Language

The first of several bioinformatic workflow management languages and systems discussed here is the Common Workflow Language (CWL; https://github.com/common-workflow-language) [8]. CWL is an open standard for describing analysis workflows and tools in a way that makes them portable and scalable across a variety of software and hardware environments, from workstations to cluster, cloud, and high-performance computing (HPC) environments. It can be applied to a number of different scientific domains including Bioinformatics, Medical Imaging, Astronomy, High Energy Physics, and Machine Learning. CWL sets itself apart from most other workflow languages by attempting to adopt open-source principles and standards such as open-stand.org, which advocates for cooperation, adherence to principles, collective empowerment, availability, and voluntary adoption. CWL is not a software, but a specification which describes command line tools and allows them to be connected together to form a workflow. CWL's commitment to creating a community which focuses on standardization and other open-source principles has led to its adoption by a number of workflow execution programs such as Toil, Arvados, Rabix, Cromwell, and Bcbio (See Tables 2.1 and 2.2). Rabix, for example, is a powerful open-source suite of tools for CWL, which include Rabix Composer, a graphical editor enabling visual programming in CWL, Rabix Benten, a language server for CWL documents, and Rabix Executor, a workflow runner that can execute CWL pipelines (https://rabix.io/). Figure 2.1 shows an example of graph generated with Rabix Composer.

The use of CWL to create tools and workflows facilitates the ease of future repeatability and reproducibility of results. This leads to greater cooperation between standard organizations,

Table 2.1 Workflow management systems

Name	Description	Website
Nextflow	Domain-specific language	http://nextflow.io
Toil	Pipeline management system	https://toil.ucsc-cgl.org
Snakemake	Domain-specific language	https://snakemake.github.io
Bpipe	Domain-specific language	http://docs.bpipe.org
WDL	Workflow specification language	https://openwdl.org/
CWL	Workflow specification language	https://www.commonwl.org/

building a foundation for collaboration. The development of CWL into a standard was made possible by adhering to five fundamental principles of standard development [13]. First, decisions regarding the direction and development of the standard must be made with equity and fairness, implementing a well-defined *due process* by which participating parties have the ability to appeal decisions made. Next, a *broad consensus* must be made in order to facilitate agreement across a range of interests. A general agreement, incorporating all views, is paramount to the establishment and persistence of an open standard. Third, activities and work being undertaken must be recorded for posterity with those records open and easily accessible to all. A consistent *transparency* must be maintained by giving advance notice of new proposals and activities. Fourth, a certain *balance* must be struck among all parties involved. No one entity involved in the development of the standard may have disproportionate influence on its direction or activities. Finally, the processes by which the standards are developed must be *open* to all. CWL stands out by encompassing all these principles and enabling cross-collaboration.

WDL: Workflow Description Language

WDL (Workflow Description Language) is a community-driven open-development workflow language developed by the Broad Institute [14]. WDL specifies data processing workflows with a human-readable and writable syntax very similarly to CWL. WDL was ostensibly developed to support Terra, a platform developed by the Broad Institute of MIT and Harvard in collaboration with Verily Life Sciences. Terra is not open-source platform and requires users to purchase credits for compute cycles. Similar to CWL, the WDL scripts are not executable and require an execution engine, such as Cromwell, MiniWDL or dxWDL, and an environment to be runnable.

NextFlow

NextFlow is a popular workflow system developed by Seqera Labs in Barcelona, Spain, designed to address numerical instability, efficient parallel execution, error tolerance, execution provenance, and traceability [9]. Similar to CWL, this domain-specific language (DSL) utilizes software containers to create scalable and reproducible workflows, enabling rapid pipeline development through the adaptation of existing pipelines written in any scripting language. NextFlow also supports GitHub and BitBucket integration, which allows for the consistent tracking of software changes and versions. Containerization, enabled by utilizing container platforms such as Docker (https://www.docker.com/) or Singularity (https://singularity.hpcng.org/), ensures numerical stability [15, 16]. It can be executed on Sun Grid Engine (SGE) (http://star.mit.edu/cluster/docs/0.93.3/guides/sge.html), Load Sharing Facility (LSF) (https://www.ibm.com/docs/en/spectrum-lsf/10.1.0), SLURM workload manager (https://slurm.schedmd.com/overview.html), Portable Batch System (PBS) (https://www.nas.nasa.gov/hecc/support/kb/portable-batch-system-(pbs)-overview_126.html) and for Kubernetes (https://kubernetes.io/), Amazon Web Services (AWS) (https://aws.amazon.com/), and Google Cloud platforms (https://cloud.google.com/) for rapid computation and

Table 2.2 Data processing platforms

Platform	Description	Website
Arvados	Open-source platform for managing, processing, and sharing genomic and other large scientific and biomedical data	https://arvados.org/
Terra	A scalable platform for biomedical research which allows data access, running analysis tools, and collaboration	https://terra.bio/
Galaxy	Open, web-based platform for accessible, reproducible, and transparent computational biomedical research.	https://github.com/galaxyproject/galaxy
bcbio-nextgen	Validated, scalable, community developed variant calling, RNA-seq, and small RNA analysis platform	https://github.com/bcbio/bcbio-nextgen
DolphinNext	A graphical user interface for distributed data processing of high-throughput genomics	https://github.com/UMMS-Biocore/dolphinnext
Sequanix	GUI for the Snakemake pipeline	https://github.com/sequana/sequana/
DNAnexus	A cloud-based data analysis and management platform for DNA sequence data	https://www.dnanexus.com/

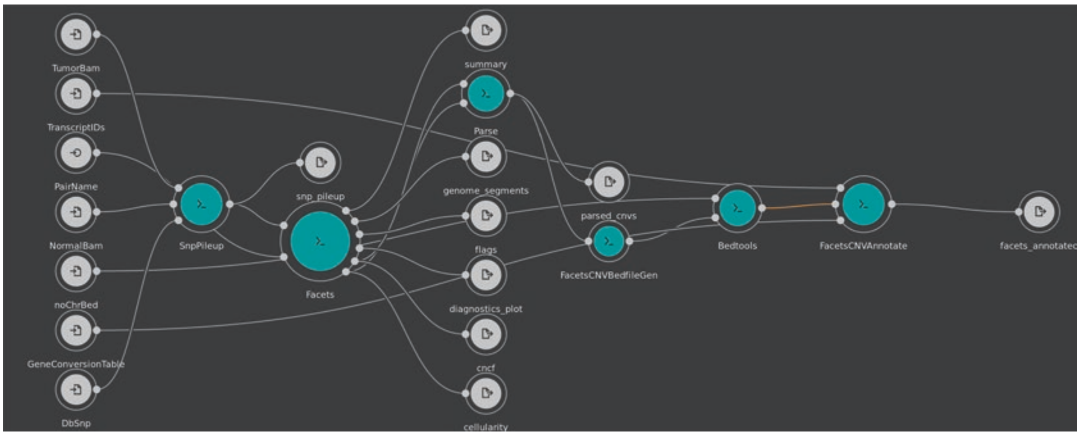


Fig. 2.1 Example of graph generated with CWL Rabix Composer

the ability to scale up projects manyfold. NextFlow also takes advantage of the "dataflow programming paradigm," where execution tasks are started automatically as soon as data is received through input channels. The Make-like approach adopted by tools such as CWL require pre-estimation of all computational dependencies as well as a directed acyclic graph (DAG). NextFlow, however, utilizes a top to bottom approach which mimics the natural flow of data.

Data Processing Platforms

The main data processing platform we will be discussing in this section is Arvados, which has been deployed in our lab and has shown great utility for our genomic processing needs. Table 2.2 summarizes the main data processing platforms.

Arvados

Arvados is a free and open-source platform for processing large volumes of genomic data [11]. This distributed computing platform for data analysis on massive data sets also enables users to share and manage their data with ease. It is licensed under the GNU Affero General Public License version 3. Two key features of Arvados are *provenance* and *reproducibility*. Arvados maintains integrity of data by recording its history and place of origin, which also reduces the incidence of replication of intermediate files. Arvados retains the history of jobs run in its infrastructure and recognizes when to re-use existing files, a cost-saving measure valuable to system administrators and informaticists alike. This is all enabled in-part by Arvados's *keep store*, a content-addressable storage system designed to run on low-cost commodity hardware or cloud services.

Other Platforms

While Arvados is free and open source, other platforms require a payment or subscription, where billing is incorporated directly into the application software.

DNAnexus (http://www.dnanexus.com/) and Terra.bio (http://terra.bio) both require the user to pay for storage and processing costs; the Galaxy project stands out with a strong, knowledgeable, and supportive online community [17] (https://usegalaxy.org/); Bcbio-nextgen is focused mainly on RNA genomic data analysis and lacks the flexibility of the other platforms mentioned in this paper [18] (https://github.com/bcbio/bcbio-nextgen); DolphinNext (https://dolphinnext.umassmed.edu/) and Sequanix (https://github.com/sequana/sequana/) are two GUIs developed specifically for Snakemake and Nextflow DSLs, respectively [19, 20]. These platforms attempt to ease the process of generating workflows by providing users with a web interface, expanding access to users with limited bioinformatics experience.

Implementation of a Precision Oncology Workflow

Designing and implementing a precision oncology pipeline requires several of the abovementioned components and entails the coordination of many tools which are then combined to create explicit workflows, relaying and processing data until it is collected and presented in a final report form. Compute and data intensive processing steps often require infrastructure consisting of large compute clusters, multiple processors, and large amounts of disc space in order to ensure reliability, efficiency, availability, and scalability. A comprehensive description of a precision oncology pipeline is provided in Chap. 1. Here, we introduce the basic syntax of CWL scripts, describe the basic steps in the design of a precision medicine workflow for DNA variant calling, and provide an overview of the software infrastructures necessary for the implementation of such workflows.

Introduction to CWL Scripting

The first step in writing a precision medicine workflow is to select the command-line tools intended for integration. This usually comprises several steps including, but not limited to, a raw read QC step, alignment, variant calling, annotation, and secondary analysis. We will use CWL as the specification for the workflow in a few examples. Figure 2.2 illustrates how inputs and outputs are isolated for reproducibility. This simple "hello world" program accepts one input parameter, writes a message to the terminal or job log, and subsequently will produce no permanent output. Several of these tools can then be written together in conjunction to form a "workflow." Figure 2.3 shows a sample workflow which extracts a java source file from a tar file and then compiles it.

There are several key considerations to make when writing and executing a workflow. First, every step in a workflow will require its own CWL description. The final inputs and outputs of the workflow are listed in the inputs and outputs section. The steps are specified under steps. The order of execution is determined by the specified connections between steps.

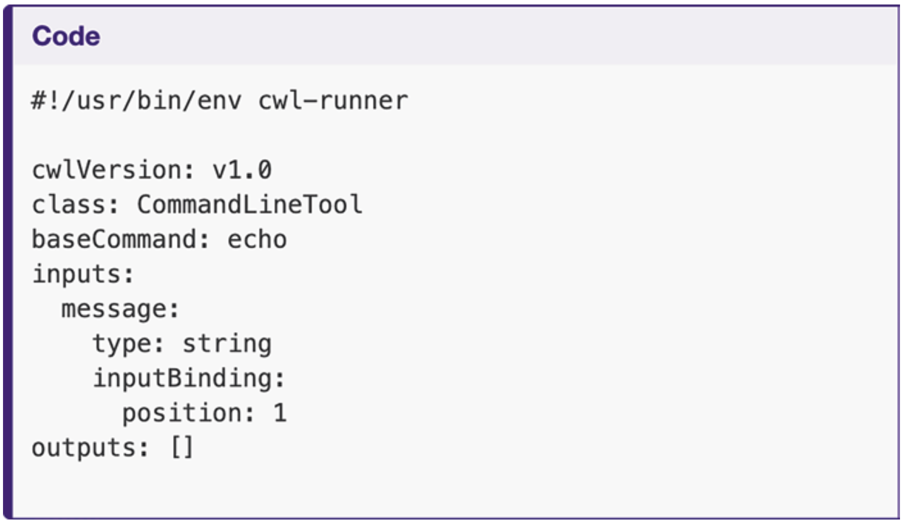
After writing the workflows, one has to choose an appropriate method for running them. In the example shown in Fig. 2.4, we use the cwl-runner. Since CWL is highly portable, the compute environment chosen to run the workflows will be up to user discretion.

Finally, Fig. 2.5 displays a more complex example of a script implementing the workflow shown in Fig. 2.1, with steps from a precision oncology pipeline which include the analysis of Copy Number Alterations (CNA) (tool: Facets [21]) and the reconstruction of tumor sub-clonal composition (tool: PhyloWGS [22]).

The Typical Steps of a Precision Oncology Pipeline

Figure 2.6 shows a typical schema for a precision oncology pipeline. After sample collection, processing and sequencing has occurred, the raw sequencing data in the form of Fastq files are used as inputs into the pipeline. Next, a series of quality control metrics are generated from the data to help determine in which areas there may be problems or poor-quality data. Metrics included in the evaluation of quality include raw sequencing data quality and depth, alignment quality, GC content, adapter contamination, and reads duplication rates [23, 24]. Evaluating these metrics allows for the identification and flagging of poor-quality data and to avoid potentially expensive and computationally intensive steps. Checking alignment quality can prevent potential false-positive single nucleotide polymorphism calls. Furthermore, it is important to verify that paired files generated from samples from the same individual, for example, normal and tumor WES samples, are indeed from the same individual, by using a tool like NGSCheckMate [25].

Next, reads are aligned to a common reference genome. Alignment algorithms such as the


Code

```
#!/usr/bin/env cwl-runner

cwlVersion: v1.0
class: CommandLineTool
baseCommand: echo
inputs:
  message:
    type: string
    inputBinding:
      position: 1
outputs: []
```

Fig. 2.2 Example of simple CWL demonstrating input/output

Burrows–Wheeler transform can be utilized to rearrange raw sequencing data and prepare it for downstream analysis and mutational calling [26]. The resulting file produced is typically a Sequence Alignment Map (SAM) or its binary version (Binary Alignment Map, BAM) file.

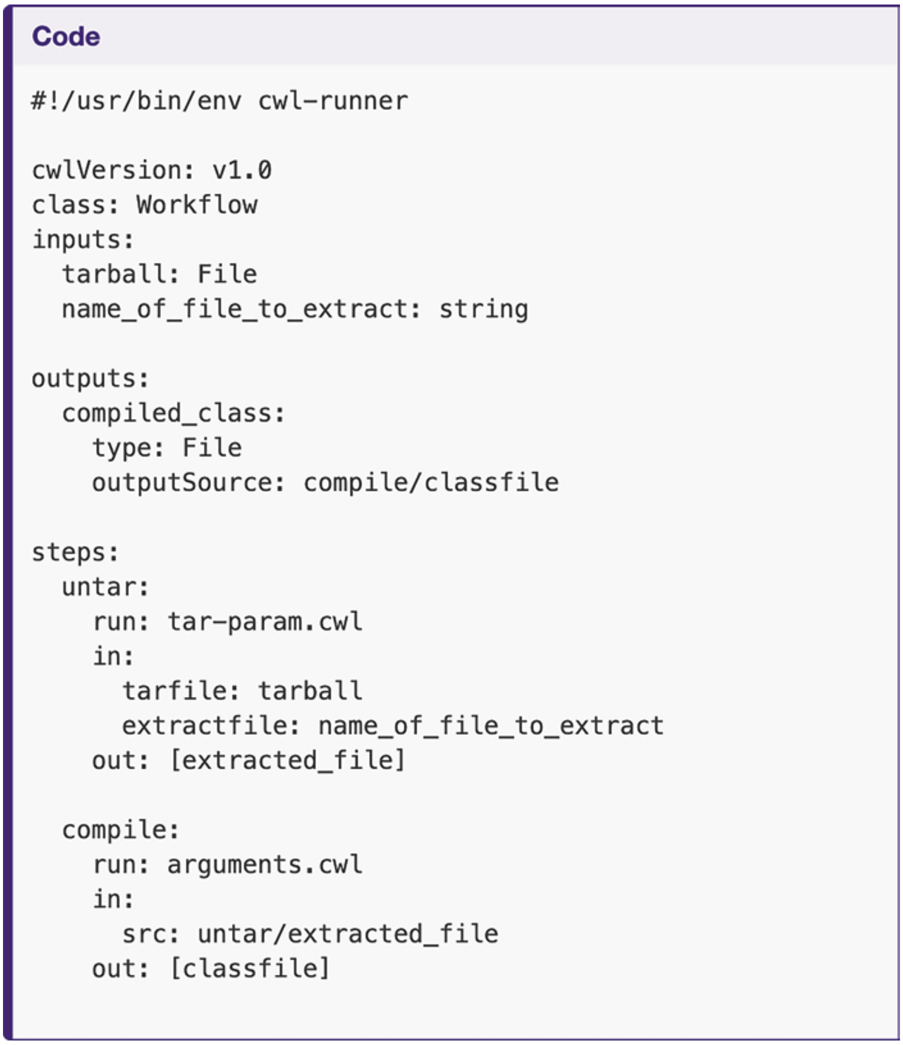
Following sequence alignment and the generation of a BAM/SAM file, a typical precision medicine pipeline would then perform variant calling by identifying where the aligned reads differ from the reference genome, producing a variant call file to be used in further downstream analysis [27] (see also Chaps. 1 and 3). After the variants have been annotated using various online databases, additional pertinent information is assigned to each variant call [28]. This information may include the definition of a variant and its genotype, basic information regarding whether it lies in a coding region, its impact on the corresponding protein (e.g., missense or synonymous mutation), or whether the variant is an insertion or a deletion. Those variants are then classified based on ACMG guidelines as pathogenic, likely pathogenic, uncertain significance, likely benign, or benign [29]. Additionally, structural variation analysis may be conducted to identify genomic alterations such as duplications, inversions, translocations, and copy number variants (CNVs) (See also Chap. 4).

The variants are then collected and classified based on whether they are actionable or not, using different databases for clinical interpretation, then summarized into reports, often after being reviewed and further annotated by pathologists [28].

In more advanced settings, the variants data can be inputted into a rule-based engine which will select and prioritize drugs matching the alterations. These "drug recommendation engines" are still in early-phase development and are typically ad hoc applications which draw on experts with domain-specific knowledge in order to auto-generate drugs with the expectation of affecting the deleterious variants in a positive manner [30–32]. Many iterations and versions of this ad hoc pipeline are being developed across academia and medical institutions for the treatment of various cancers. Each pipeline with its own unique set of rules and considerations based on the model-disease specifications.

Software Infrastructures for Precision Oncology Platforms

Here we provide some background on the software infrastructure for a precision oncology pipeline. The diagram in Fig. 2.7 illustrates the



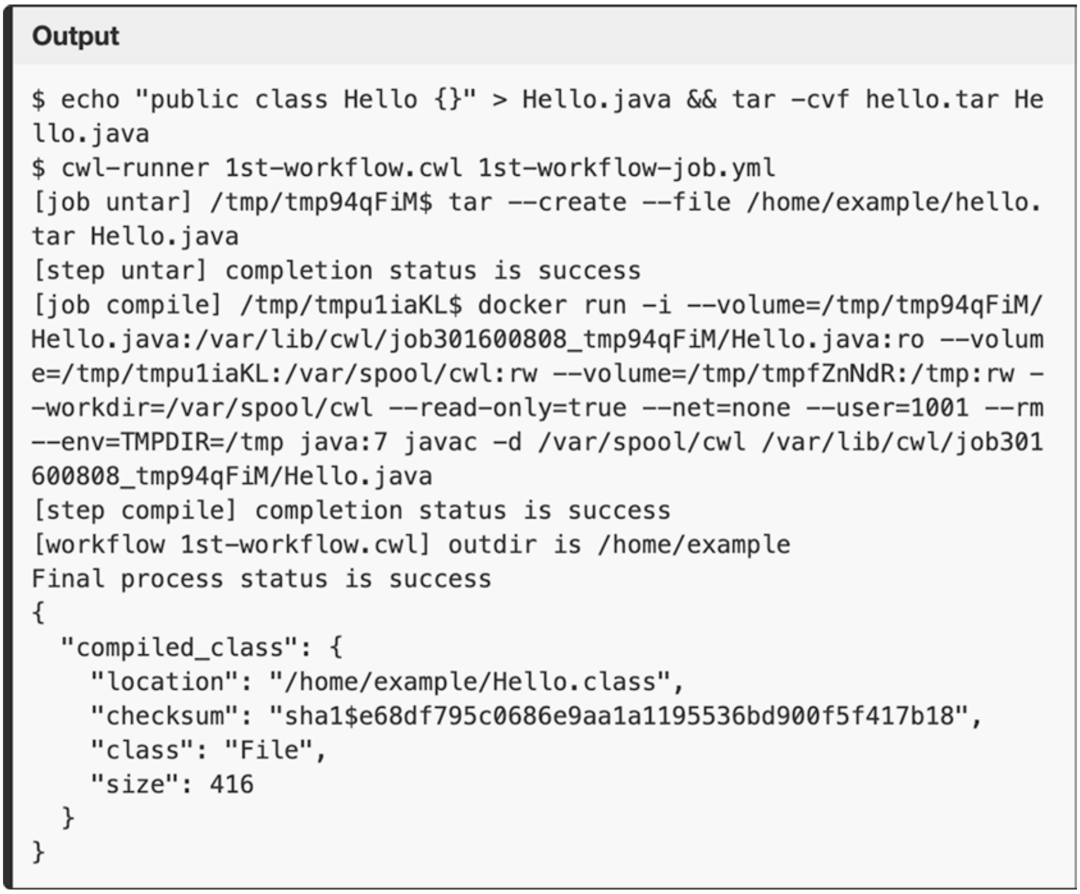
Code

```
#!/usr/bin/env cwl-runner

cwlVersion: v1.0
class: Workflow
inputs:
  tarball: File
  name_of_file_to_extract: string

outputs:
  compiled_class:
    type: File
    outputSource: compile/classfile

steps:
  untar:
    run: tar-param.cwl
    in:
      tarfile: tarball
      extractfile: name_of_file_to_extract
    out: [extracted_file]

  compile:
    run: arguments.cwl
    in:
      src: untar/extracted_file
    out: [classfile]
```

Fig. 2.3 Example of CWL workflow which extracts a java source file from a tar file and compiles it

components which comprise the Arvados technical architecture. It can be deployed locally, or on a number of different cloud providers such as Amazon Web Services (AWS) (https://aws.amazon.com/), the Google Cloud Platform (GCP) (https://cloud.google.com/), or on Microsoft Azure (https://azure.microsoft.com/). Several key components work together in harmony to create an elastic computing environment where the overall resource footprint available or consumed by a specific job can grow or shrink on demand. The ability of Arvados to quickly expand or decrease computer processing, memory, and storage resources as well as manage data through a content-addressable distributed storage system sets it apart from its competitors. These components are the container orchestration system called “Crunch,” the distributed storage system “Keep,” the REST API Server, the CLI, the GUI “Workbench,” native language SDKs, Data Manager, Node Manager, and Keep proxy.

The main two innovations of the Arvados platform are “Crunch” and “Keep.” The Crunch container orchestration management engine executes



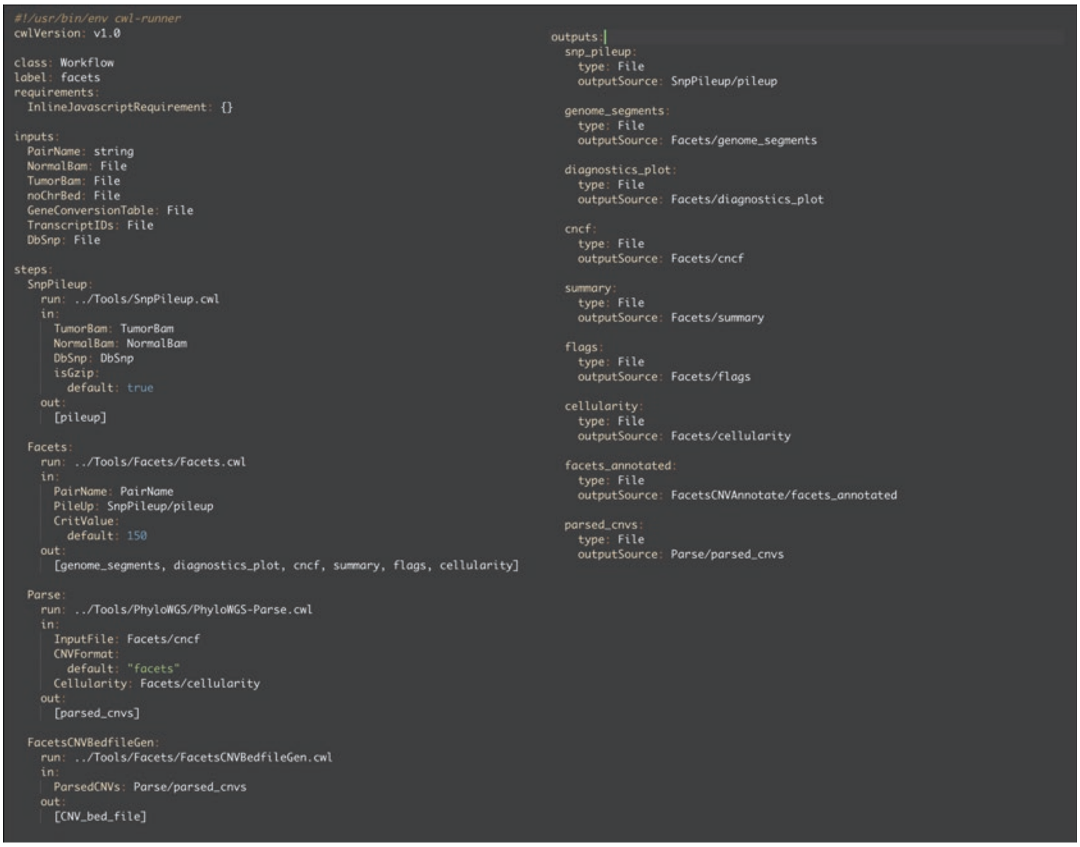
Output

```
$ echo "public class Hello {}" > Hello.java && tar -cvf hello.tar He
llo.java
$ cwl-runner 1st-workflow.cwl 1st-workflow-job.yml
[job untar] /tmp/tmp94qFiM$ tar --create --file /home/example/hello.
tar Hello.java
[step untar] completion status is success
[job compile] /tmp/tmpu1iaKL$ docker run -i --volume=/tmp/tmp94qFiM/
Hello.java:/var/lib/cwl/job301600808_tmp94qFiM/Hello.java:ro --volum
e=/tmp/tmpu1iaKL:/var/spool/cwl:rw --volume=/tmp/tmpfZnNdR:/tmp:rw -
-workdir=/var/spool/cwl --read-only=true --net=none --user=1001 --rm
--env=TMPDIR=/tmp java:7 javac -d /var/spool/cwl /var/lib/cwl/job301
600808_tmp94qFiM/Hello.java
[step compile] completion status is success
[workflow 1st-workflow.cwl] outdir is /home/example
Final process status is success
{
  "compiled_class": {
    "location": "/home/example/Hello.class",
    "checksum": "sha1$e68df795c0686e9aa1a1195536bd900f5f417b18",
    "class": "File",
    "size": 416
  }
}
```

Fig. 2.4 Example of cwl-runner execution

CWLs while maintaining provenance and reproducibility. It accomplishes this by automatically tracking the origin of result data; therefore, it is able to compare workflows to one another, avoiding the need to repeat previously performed data analysis. This saves on cost and time, two significant considerations when executing a workflow or data analysis. Crunch also provides the ability to scale horizontally by provisioning compute nodes upon demand, delivering cost-effective performance. Finally, the Crunch engine isolates workloads by running jobs inside of Docker containers, a standard unit of software that packages up code and all its dependencies [15].

The Keep system efficiently handles data storage and management using a content-addressable distributed storage system. It is able to handle petabyte-sized data sets, scaling accordingly by utilizing location-addressed storage. A permanent universally unique identifier (UUID) is then given to each content address. This creates a highly scalable flat address space, virtualizing storage access. The benefits of the keep store system include, elimination of duplication, canonical records, provenance, easy management of temporary data, flexible organization, high reliability, security and access control, POSIX interface, data sharing, and versioning.



```
#!/usr/bin/env cwl-runner
cwlVersion: v1.0

class: Workflow
label: facets
requirements:
  InlineJavascriptRequirement: {}

inputs:
  PairName: string
  NormalBam: File
  TumorBam: File
  noChrBed: File
  GeneConversionTable: File
  TranscriptIDs: File
  DbSnp: File

steps:
  SnpPileup:
    run: ../Tools/SnpPileup.cwl
    in:
      TumorBam: TumorBam
      NormalBam: NormalBam
      DbSnp: DbSnp
      isGzip:
        default: true
    out:
      [pileup]

  Facets:
    run: ../Tools/Facets/Facets.cwl
    in:
      PairName: PairName
      PileUp: SnpPileup/pileup
      CritValue:
        default: 150
    out:
      [genome_segments, diagnostics_plot, cncf, summary, flags, cellularity]

  Parse:
    run: ../Tools/PhyloWGS/PhyloWGS-Parse.cwl
    in:
      InputFile: Facets/cncf
      CNVFormat:
        default: "facets"
      Cellularity: Facets/cellularity
    out:
      [parsed_cnvs]

  FacetsCNVBedfileGen:
    run: ../Tools/Facets/FacetsCNVBedfileGen.cwl
    in:
      ParsedCNVs: Parse/parsed_cnvs
    out:
      [CNV_bed_file]
```

```
outputs:
  snp_pileup:
    type: File
    outputSource: SnpPileup/pileup

  genome_segments:
    type: File
    outputSource: Facets/genome_segments

  diagnostics_plot:
    type: File
    outputSource: Facets/diagnostics_plot

  cncf:
    type: File
    outputSource: Facets/cncf

  summary:
    type: File
    outputSource: Facets/summary

  flags:
    type: File
    outputSource: Facets/flags

  cellularity:
    type: File
    outputSource: Facets/cellularity

  facets_annotated:
    type: File
    outputSource: FacetsCNVAnnotate/facets_annotated

  parsed_cnvs:
    type: File
    outputSource: Parse/parsed_cnvs
```

Fig. 2.5 Example of a CWL script from a precision oncology pipeline. The script defines the step to run a CNV analysis using the tool Facets. The class field indicates that this document describes a command line tool. The three main sections describe the inputs, steps, and outputs of the pipeline

The installation and deployment of such infrastructures can be accomplished on GNU/Linux systems either bare metal, or on AWS, GCP, and Azure cloud services. The multi-host installation provides the highest throughput and can be accomplished using Salt, an automated infrastructure management software [26]. The Arvados salt formula can be found at https://github.com/saltstack-formulas/arvados-formula.git, and the steps for deployment are as follows:

1. Fork/copy the formula to your Salt master host.
2. Edit the Arvados, nginx, postgres, locale, and docker pillars to match your desired configuration.
3. Run a state.apply to get it deployed.

After this step, the cloud/software engineer will then need to set up the DNS in order to access the cluster's nodes. Typical operations include running a workflow, uploading, and downloading data from keep. Periodically, Arvados releases new versions of the platform which will require a short maintenance window where data processing will need to be suspended.

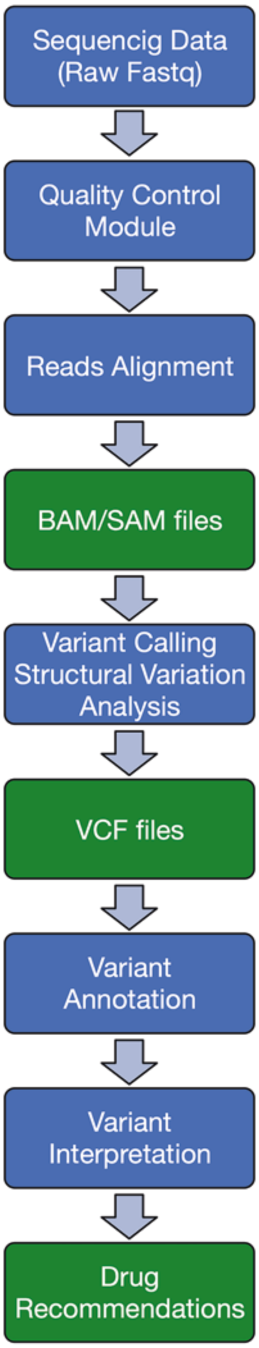


Fig. 2.6 A typical schema of a precision oncology pipeline

Conclusion

Utilizing an appropriate domain-specific language for workflow development and execution is a necessity. The adept bioinformatics engineer/analyst will require the combination of many tools and that combination will need to be seamless. CWL, WDL, Snakemake, and NextFlow all provide the portability and flexibility needed for precision oncology workflows. When the requisite components for a robust pipeline are in place, the effort to scale up your workload will be minimal.

Although many workflow systems are available, we have found that the combination of CWL and Arvados serve for the most comprehensive platform for genomics data processing at large scale. CWL's requirements for explicitness and isolation lead to more flexibility, portability, and scalability for your workloads. With a large user base, CWL is and will continue to be supported and updated on a regular basis. This will ensure the resilience and longevity of pipelines and precision medicine platforms.

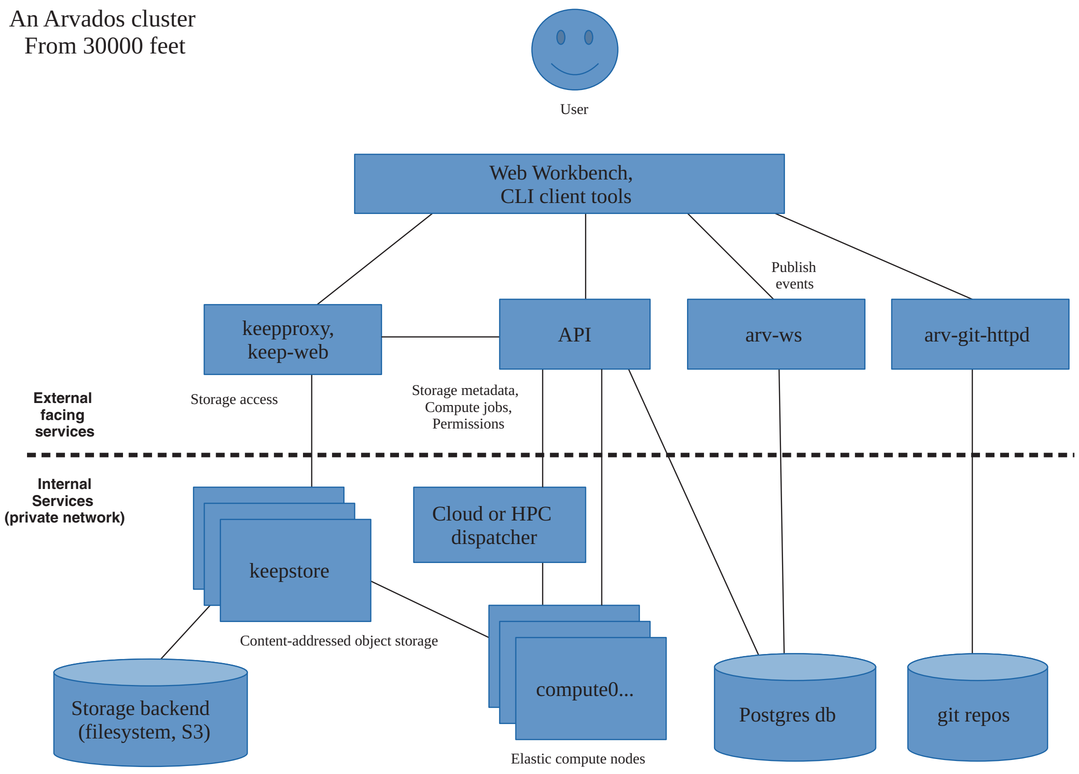


Fig. 2.7 The Arvados technical architecture

References

1. Laganà A, et al. Precision medicine for relapsed multiple myeloma on the basis of an integrative multiomics approach. JCO Precis Oncol. 2018;2018:1–17.
2. Berger MF, Mardis ER. The emerging clinical relevance of genomics in cancer medicine. Nat Rev Clin Oncol. 2018;15:353–65.
3. Johnson TM. Perspective on precision medicine in oncology. Pharmacotherapy. 2017;37:988–9.
4. Odle TG. Precision medicine in breast cancer. Radiol Technol. 2017;88:401M–21M.
5. Bødker JS, et al. Development of a precision medicine workflow in hematological cancers, Aalborg University Hospital, Denmark. Cancers (Basel). 2020;12:312.
6. Jäger N. Bioinformatics workflows for clinical applications in precision oncology. In: Seminars in cancer biology. Academic Press; 2021. https://doi.org/10.1016/j.semcancer.2020.12.020.
7. Altintas I, et al. Understanding collaborative studies through interoperable workflow provenance. In: Lecture notes in computer science. Berlin Heidelberg: Springer; 2010. p. 42–58.
8. Amstutz P, et al. Common workflow language, v1. 0. 2016.
9. Di Tommaso P, et al. Nextflow enables reproducible computational workflows. Nat Biotechnol. 2017;35:316–9.
10. Mölder F, et al. Sustainable data analysis with Snakemake. F1000Res. 2021;10:33.
11. Amstutz P. Portable, reproducible analysis with arvados. F1000Res. 2015;4
12. Terra. https://terra.bio/
13. The Modern Standards Paradigm – 5 Key Principles. https://open-stand.org/about-us/principles/
14. Workflow Description Language (WDL). *OpenWDL* https://openwdl.org/
15. Boettiger C. An introduction to Docker for reproducible research. Oper Syst Rev. 2015;49:71–9.
16. Kurtzer GM, Sochat V, Bauer MW. Singularity: scientific containers for mobility of compute. PLoS One. 2017;12:e0177459.
17. Blankenberg D, Hillman-Jackson J. Analysis of next-generation sequencing data using Galaxy. Methods Mol Biol. 2014;1150:21–43.
18. Guimera RV. bcbio-nextgen: automated, distributed next-gen sequencing pipeline. EMBnet J. 2012;17:30.
19. Yukselen O, Turkyilmaz O, Ozturk AR, Garber M, Kucukural A. DolphinNext: a distributed data processing platform for high throughput genomics. BMC Genomics. 2020;21:310.

20. Desvillechabrol D, et al. Sequanix: a dynamic graphical interface for Snakemake workflows. Bioinformatics. 2018;34:1934–6.
21. Shen R, Seshan VE. FACETS: allele-specific copy number and clonal heterogeneity analysis tool for high-throughput DNA sequencing. Nucleic Acids Res. 2016;44:e131.
22. Deshwar AG, et al. PhyloWGS: reconstructing subclonal composition and evolution from whole-genome sequencing of tumors. Genome Biol. 2015;16:35.
23. Andrews S, et al. FastQC: a quality control tool for high throughput sequence data. 2010.
24. Chen S, Zhou Y, Chen Y, Gu J. fastp: an ultra-fast all-in-one FASTQ preprocessor. Bioinformatics. 2018;34:i884–90.
25. Lee S, et al. NGSCheckMate: software for validating sample identity in next-generation sequencing studies within and across data types. Nucleic Acids Res. 2017;45:e103.
26. Li H, Durbin R. Fast and accurate short read alignment with Burrows–Wheeler transform. Bioinformatics. 2009;25:1754–60.
27. Koboldt DC. Best practices for variant calling in clinical sequencing. Genome Med. 2020;12:91.
28. Li X, Warner JL. A review of precision oncology knowledgebases for determining the clinical actionability of genetic variants. Front Cell Dev Biol. 2020;8:48.
29. Li MM, et al. Standards and guidelines for the interpretation and reporting of sequence variants in cancer: a joint consensus recommendation of the Association for Molecular Pathology, American Society of Clinical Oncology, and College of American Pathologists. J Mol Diagn. 2017;19:4–23.
30. Piñeiro-Yáñez E, et al. PanDrugs: a novel method to prioritize anticancer drug treatments according to individual genomic data. Genome Med. 2018;10(1):1–11.
31. Yu Y, et al. PreMedKB: an integrated precision medicine knowledgebase for interpreting relationships between diseases, genes, variants and drugs. Nucleic Acids Res. 2019;47:D1090–101.
32. Xu Q, et al. OncoPDSS: an evidence-based clinical decision support system for oncology pharmacotherapy at the individual level. BMC Cancer. 2020;20:740.