Snakemake

Workflow Management System

Guía básica para entender el sistema.

Felipe Betancourt Figueroa.

02 de diciembre de 2020

Introducción:

El sistema de gestión de flujo de trabajo Snakemake es una herramienta para crear análisis de datos reproducibles y escalables. Los flujos de trabajo se describen a través de un lenguaje basado en Python legible por humanos. Se pueden escalar sin problemas a entornos de servidor, clúster, red y nube, sin la necesidad de modificar la definición del flujo de trabajo. Finalmente, los flujos de trabajo de Snakemake pueden implicar una descripción del software requerido, que se implementará automáticamente en cualquier entorno de ejecución.

Software necesario:

- Python https://www.python.org/
- Miniconda https://conda.io/en/latest/miniconda.html

Instalación:

- Se recomienda instalar conda para reemplazar el solver default de conda
    - $ conda install -c conda-forge mamba
- Snakemake Full (Para sistemas Unix[Linux/MacOS])
    - $ mamba create -c conda-forge -c bioconda -n snakemake snakemake
- Snakemake Minimal (Para sistemas Windows)
    - $ mamba create -c bioconda -c conda-forge -n snakemake snakemake-minimal
- Esto instalará Snakemake en un entorno de software aislado, que debe activarse con:
    - $ conda activate snakemake
    - $ snakemake –help
- La instalación en entornos aislados es la mejor práctica para evitar efectos secundarios con otros paquetes.

Snakefile:

- En Snakemake, los flujos de trabajo se especifican como Snakefiles.
- Inspirado por GNU Make, un Snakefile contiene reglas que indican cómo crear archivos de salida a partir de archivos de entrada.
- Las dependencias entre reglas se manejan implícitamente, haciendo coincidir los nombres de los archivos de entrada con los de salida. Por lo tanto, se pueden usar wildcards para escribir reglas generales.
- Gramática:
    - La sintaxis del Snakefile obedece a la siguiente gramática, dada en forma extendida Backus-Naur (EBNF)

```
snakemake  = statement | rule | include | workdir
rule       = "rule" (identifier | "") ":" ruleparams
include    = "include:" stringliteral
workdir    = "workdir:" stringliteral
ni         = NEWLINE INDENT
ruleparams = [ni input] [ni output] [ni params] [ni message] [ni threads] [ni (run | shell)]
NEWLINE snakemake
input      = "input" ":" parameter_list
output     = "output" ":" parameter_list
params     = "params" ":" parameter_list
log        = "log" ":" parameter_list
benchmark  = "benchmark" ":" statement
cache      = "cache" ":" bool
message    = "message" ":" stringliteral
threads    = "threads" ":" integer
resources  = "resources" ":" parameter_list
version    = "version" ":" statement
run        = "run" ":" ni statement
shell      = "shell" ":" stringliteral
conda      = "conda" ":" parameter_list
configfile = "configfile" ":" stringliteral
```

- Estructura básica de un Snakefile:

```
rule all:
    input:
        "dir/outFile1.json"
rule etl:
    input:
        "dir/ontology.owl"
    output:
        "dir/outFile1.json",
        "dir/outFile2.log"
    conda:
        "dir/dependencies.yaml"
    shell:
        "python etl.py –i {input} –o {output[0]} –l {output[1]}"
```

Archivo de configuración y de dependencias:

- Snakemake apoya directamente la configuración de su flujo de trabajo. Se proporciona una configuración como un archivo JSON o YAML y se puede cargar con:
    - configfile: "path/to/config.json"
- El archivo de configuración se puede utilizar para definir un diccionario de parámetros de configuración y sus valores. En el flujo de trabajo, la configuración es accesible a través de la configuración de variable global. Ejemplo:

```
rule all:
  input:
   expand("{sample}.{param}.output.pdf",sample=config["samples"], param=conf
ig["yourparam"])
```

- En ocasiones requerimos de ejecutar scripts que dependan de ciertas bibliotecas, o de una versión anterior de Python, cosa que nos complica el ejecutar nuestro Snakefile en otro dispositivo, conda nos facilita instalar de manera aislada las dependencias que necesitemos.
    - En el Snakefile:

```
conda:
       "dir/dependencies.yaml"
```

    - Archivo .yaml:

```
channels: #Los canales que utilizará conda para descargar las dependencias y
 crear el entorno aislado
  - bioconda
  - conda-forge
dependencies: #Las bibliotecas o software que se require para los procesos s
e puede poner la version de los mismos
  - python = 2.7
  - owlready2
```

    - En la línea de comandos usamos la opción --use-conda para que se utilicen las dependencias den el archivo .yaml:

```
$ snakemake --use-conda
```

Interfaz de línea de comandos:

```
$ snakemake
$ snakemake –s “mySnakefile"
$ snakemake –n
$ snakemake –r
$ snakemake –f
$ snakemake --cores 4
$ snakemake –R "myRule"
$ snakemake --use–conda
$ snakemake –report
```

Todas las Opciones:

```
usage: snakemake [-h] [--dry-run] [--profile PROFILE]
                 [--cache [RULE [RULE ...]]] [--snakefile FILE] [--cores [N]]
                 [--local-cores N] [--resources [NAME=INT [NAME=INT ...]]]
                 [--set-threads RULE=THREADS [RULE=THREADS ...]]
                 [--set-scatter NAME=SCATTERITEMS [NAME=SCATTERITEMS ...]]
                 [--default-resources [NAME=INT [NAME=INT ...]]]
                 [--preemption-default PREEMPTION_DEFAULT]
                 [--preemptible-rules PREEMPTIBLE_RULES [PREEMPTIBLE_RULES ...]]
                 [--config [KEY=VALUE [KEY=VALUE ...]]]
                 [--configfile FILE [FILE ...]]
                 [--envvars VARNAME [VARNAME ...]] [--directory DIR] [--touch]
                 [--keep-going] [--force] [--forceall]
                 [--forcerun [TARGET [TARGET ...]]]
                 [--prioritize TARGET [TARGET ...]]
                 [--batch RULE=BATCH/BATCHES] [--until TARGET [TARGET ...]]
                 [--omit-from TARGET [TARGET ...]] [--rerun-incomplete]
```

```
                  [--shadow-prefix DIR] [--scheduler [{ilp,greedy}]]
                  [--wms-monitor [WMS_MONITOR]]
                  [--scheduler-ilp-solver {COIN_CMD}] [--no-subworkflows]
                  [--groups GROUPS [GROUPS ...]]
                  [--group-components GROUP_COMPONENTS [GROUP_COMPONENTS ...]]
                  [--report [FILE]] [--report-stylesheet CSSFILE]
                  [--edit-notebook TARGET] [--notebook-listen IP:PORT]
                  [--lint [{text,json}]] [--generate-unit-tests [TESTPATH]]
                  [--export-cwl FILE] [--list] [--list-target-rules] [--dag]
                  [--rulegraph] [--filegraph] [--d3dag] [--summary]
                  [--detailed-summary] [--archive FILE]
                  [--cleanup-metadata FILE [FILE ...]] [--cleanup-shadow]
                  [--skip-script-cleanup] [--unlock] [--list-version-changes]
                  [--list-code-changes] [--list-input-changes]
                  [--list-params-changes] [--list-untracked]
                  [--delete-all-output] [--delete-temp-output]
                  [--bash-completion] [--keep-incomplete] [--drop-metadata]
                  [--version] [--reason] [--gui [PORT]] [--printshellcmds]
                  [--debug-dag] [--stats FILE] [--nocolor] [--quiet]
                  [--print-compilation] [--verbose] [--force-use-threads]
                  [--allow-ambiguity] [--nolock] [--ignore-incomplete]
                  [--max-inventory-time SECONDS] [--latency-wait SECONDS]
                  [--wait-for-files [FILE [FILE ...]]] [--notemp]
                  [--keep-remote] [--keep-target-files]
                  [--allowed-rules ALLOWED_RULES [ALLOWED_RULES ...]]
                  [--max-jobs-per-second MAX_JOBS_PER_SECOND]
                  [--max-status-checks-per-second MAX_STATUS_CHECKS_PER_SECOND]
                  [-T RESTART_TIMES] [--attempt ATTEMPT]
                  [--wrapper-prefix WRAPPER_PREFIX]
                  [--default-remote-provider
{S3,GS,FTP,SFTP,S3Mocked,gfal,gridftp,iRODS,AzBlob}]
                  [--default-remote-prefix DEFAULT_REMOTE_PREFIX]
                  [--no-shared-fs] [--greediness GREEDINESS] [--no-hooks]
                  [--overwrite-shellcmd OVERWRITE_SHELLCMD] [--debug]
                  [--runtime-profile FILE] [--mode {0,1,2}]
                  [--show-failed-logs] [--log-handler-script FILE]
                  [--log-service {none,slack}]
                  [--cluster CMD | --cluster-sync CMD | --drmaa [ARGS]]
                  [--cluster-config FILE] [--immediate-submit]
                  [--jobscript SCRIPT] [--jobname NAME]
                  [--cluster-status CLUSTER_STATUS] [--drmaa-log-dir DIR]
                  [--kubernetes [NAMESPACE]] [--container-image IMAGE]
                  [--tibanna] [--tibanna-sfn TIBANNA_SFN]
                  [--precommand PRECOMMAND]
                  [--tibanna-config TIBANNA_CONFIG [TIBANNA_CONFIG ...]]
                  [--google-lifesciences]
                  [--google-lifesciences-regions GOOGLE_LIFESCIENCES_REGIONS
[GOOGLE_LIFESCIENCES_REGIONS ...]]
                  [--google-lifesciences-location GOOGLE_LIFESCIENCES_LOCATION]
                  [--google-lifesciences-keep-cache] [--tes URL] [--use-conda]
                  [--list-conda-envs] [--conda-prefix DIR]
                  [--conda-cleanup-envs]
                  [--conda-cleanup-pkgs [{tarballs,cache}]]
                  [--conda-create-envs-only] [--conda-frontend {conda,mamba}]
                  [--use-singularity] [--singularity-prefix DIR]
                  [--singularity-args ARGS] [--use-envmodules]
                  [target [target ...]]
```

Ejecución ver en documentación:

https://snakemake.readthedocs.io/en/stable/executing/cli.html#EXECUTION