Organizing software and actions: Argo Workflow

Alessandro Costantini

alessandro.costantini@cnaf.infn.it

Summary

- Software&Actions: VCS
- Workflows
- CI/CD
- Working with VCS: Git
  - Basic concepts
- CI/CD in GitLab
- GitHub Actions
- Argo

Organizing SW and Actions

- Version control — also known as source control or revision control — is an important software development **practice for tracking and managing changes** made to code and other files. It is closely related to source code management.
- With version control, every change made to the code is tracked. This allows developers to see the entire **history of who changed what** at any given time — and **roll back** to an earlier version if they need to.
  - If developers code concurrently and create **incompatible changes**, version control identifies the problem areas so that team **members can quickly** revert changes to a previous version, compare changes, or identify who committed the problem code through the revision history

Benefits of version control

- **Quality**
Teams can review, comment, and improve each other’s code and assets.

- **Acceleration**
Branch code, make changes, and merge commits faster.

- **Visibility**
Understand and improve team collaboration to foster greater release build and release patterns.

- A version control system (VCS) tracks changes to a file or set of files over time.

Version Control Systems

Git

open source distributed system that is used for software projects of any size, making it a popular option for startups, enterprise, and everything in between.

Subversion

This system keeps all of a project's files on a single codeline making it impossible to branch, so it's easy to scale for large projects. It's simple to learn and features folder security measures, so access to subfolders can be restricted.

Mercurial

The system enables rapid scaling and collaborative development, with an intuitive interface. The flexible command line interface enables users to begin using the system immediately.

Version Control Systems

Git

open source distributed system that is used for software projects of any size, making it a popular option for startups, enterprise, and everything in between.

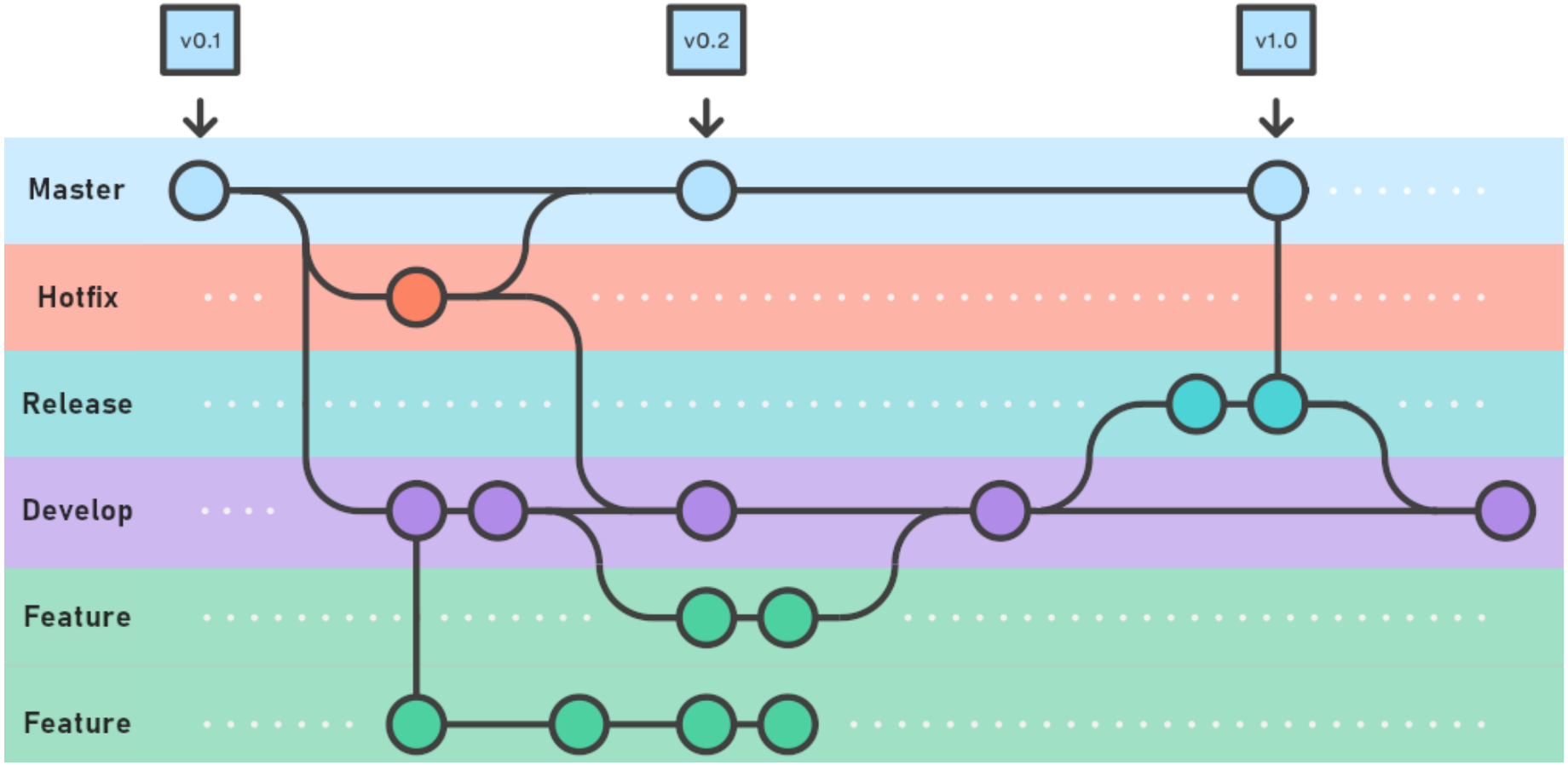
Subversion

This system keeps all of a project's files on a single codeline making it impossible to branch, so it's easy to scale for large projects. It's simple to learn and features folder security measures, so access to subfolders can be restricted.

Mercurial

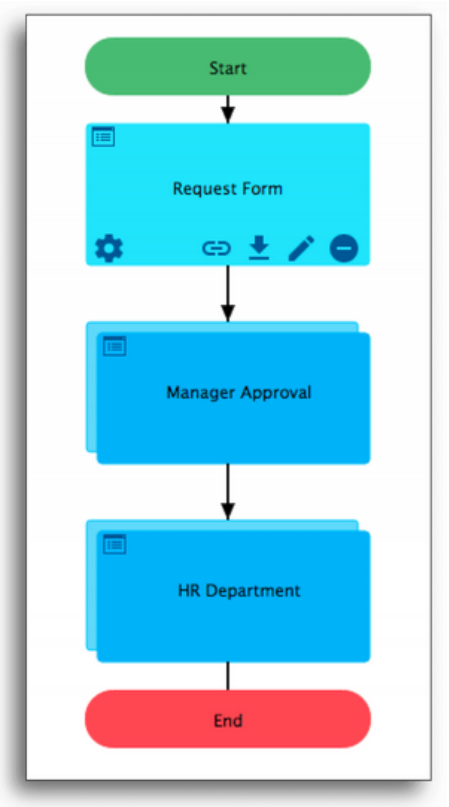
The system enables rapid scaling and collaborative development, with an intuitive interface. The flexible command line interface enables users to begin using the system immediately.

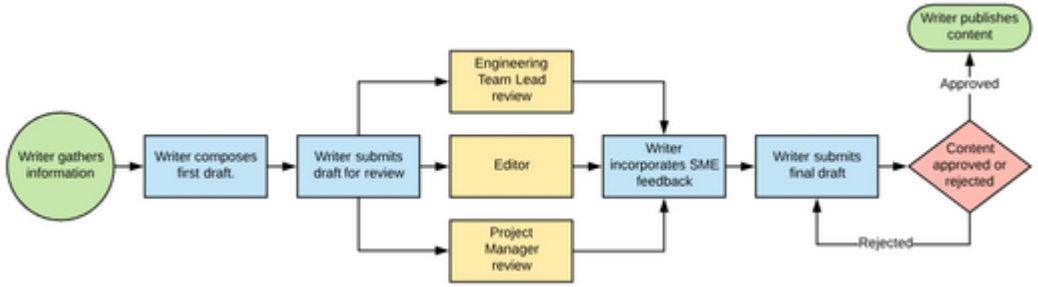
Git workflows



Workflows

- A Workflow is defined as a **sequence of tasks** that processes a set of data through a specific path from initiation to completion



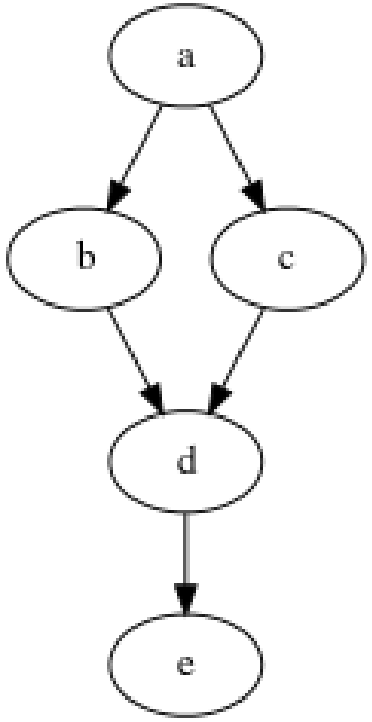


Workflow management

- Software that helps us to **manage the documents and processes**
- **It helps us to…**
  - Automate the process
  - Follow up on pending tasks
  - Get the picture and the state of the workflow
  - Manage the action

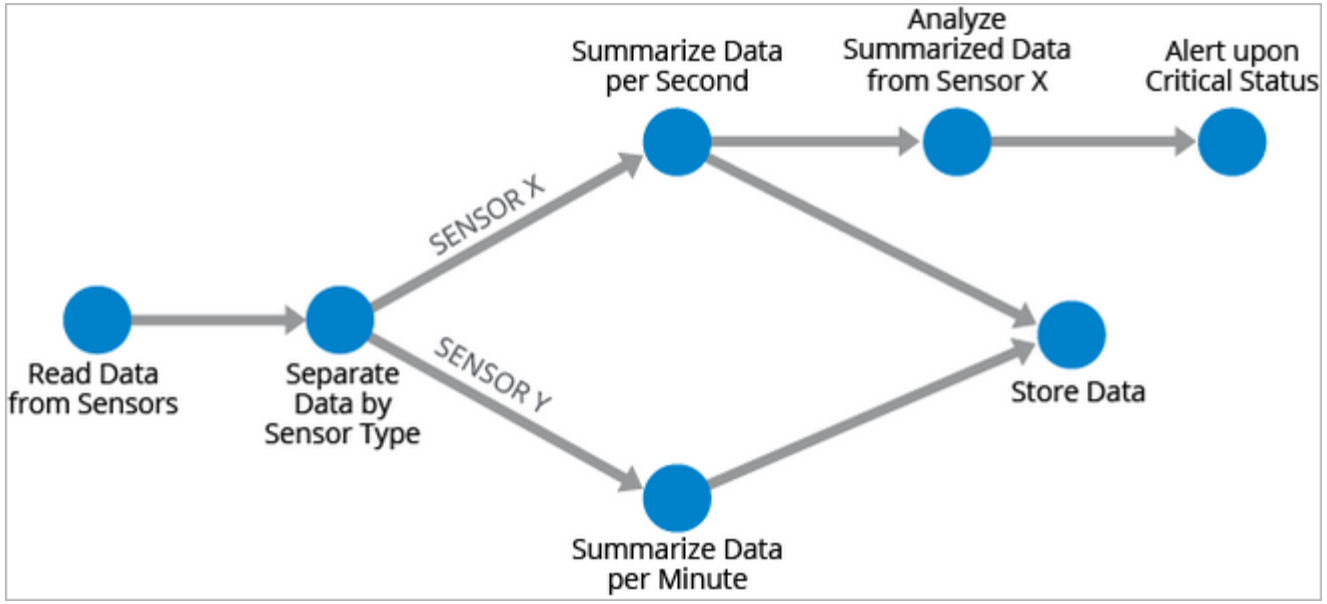
A bit about DAG Pipelines

- A **directed acyclic graph (DAG)** is a conceptual representation of a series of activities.
- The order of the **activities** is depicted by a graph, which is visually presented as a set of **circles**, each one representing an activity, some of which are **connected by lines**, which represent the flow from one activity to another.
- Each circle is known as a "**vertex**" and each line is known as an "**edge**."
- "**Directed**" means that each edge has a **defined** direction
- "**Acyclic**" means that there are **no loops**



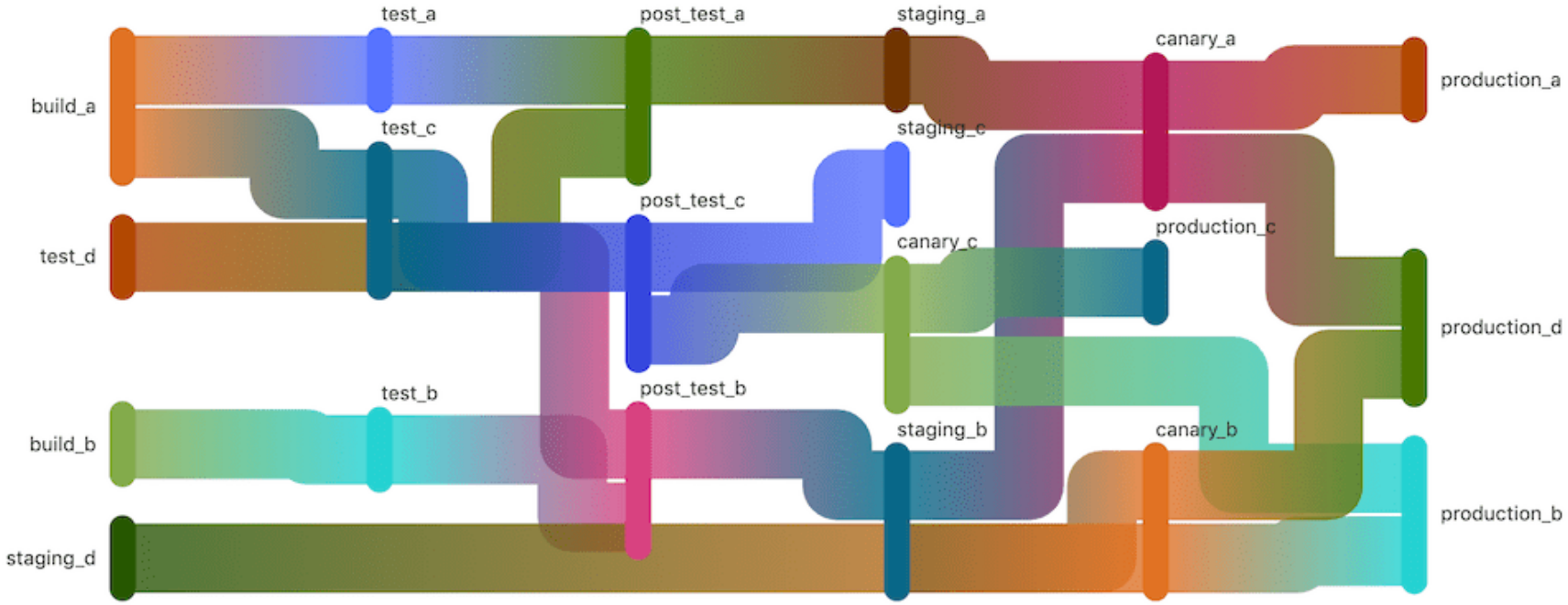
Why Are DAG Useful?

- DAGs are useful for representing many different types of flows
  - Including data processing flows



Why Are DAG Useful?

- Can be used in the context of a **software organization pipeline** (CI/CD) to build relationships between jobs such that execution is performed in the quickest possible manner, regardless how stages may be set up.



CI/CD

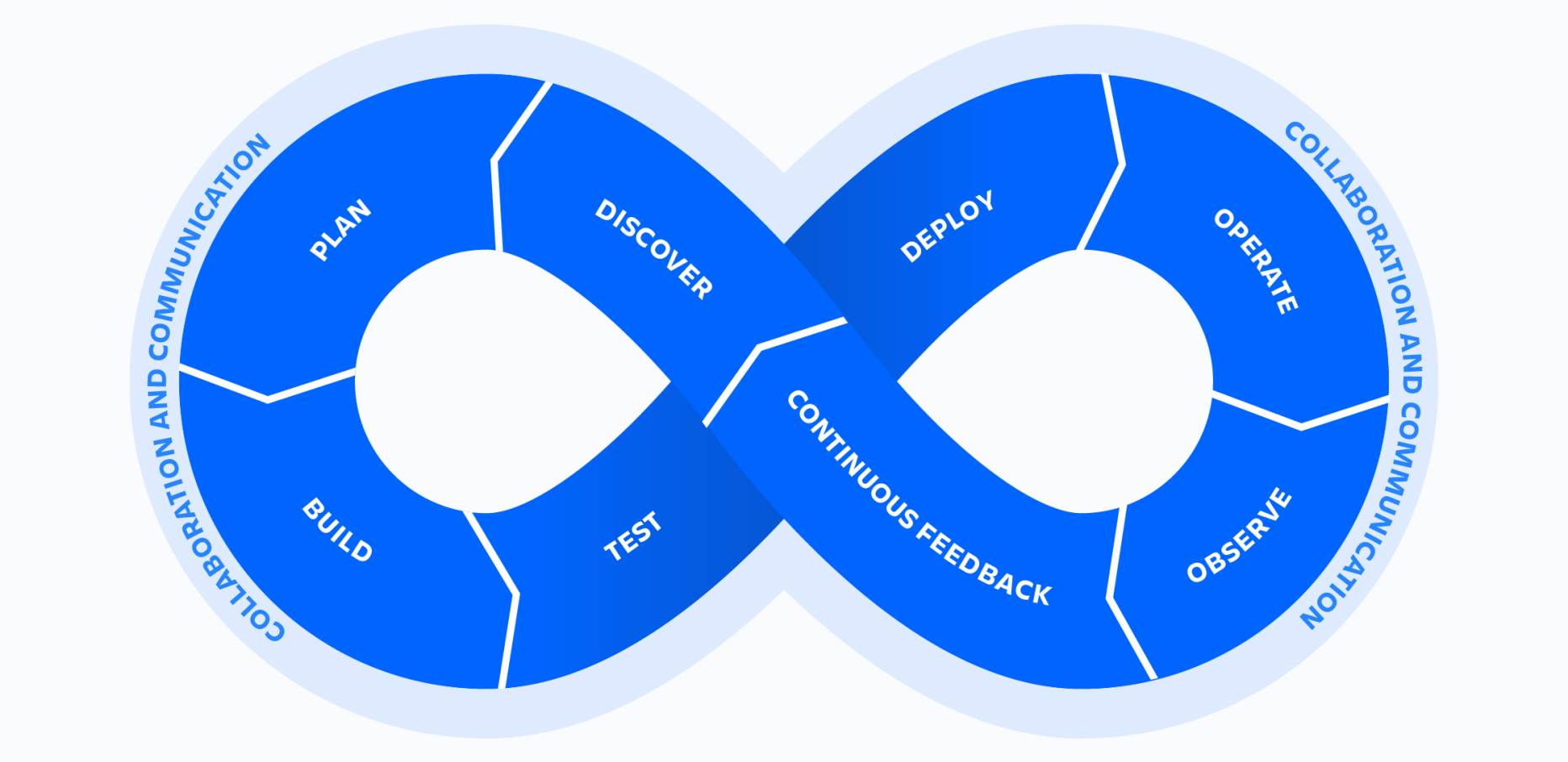
CI/CD

- CI/CD (continuous integration/continuous delivery) are the steps to be executed to **provide a new version of the software**.
- A **pipeline** of CI/CD are **procedure used to optimize software** provisioning through a **DevOps** o Site Reliability Engineering (SRE) approach.
- The CI/CD flux introduce both **monitoring** and **automation** aimed to optimize the process bringing to the development of the applications
  - Integration phase
  - testing
  - distribution
  - deployment
- The main advantage of the CI/CD is on the **procedure automation**

DevOps

- DevOps is a set of practices, tools, and a cultural philosophy that automate and integrate the processes between software development and IT teams.
- The DevOps movement began around 2007
  - **Traditional** software development model
  - Developers who wrote code worked apart from operations who deployed and supported the code
- The term **DevOps**, a combination of the words **development** and **operations**, reflects the process of **integrating** these disciplines into one, continuous process.

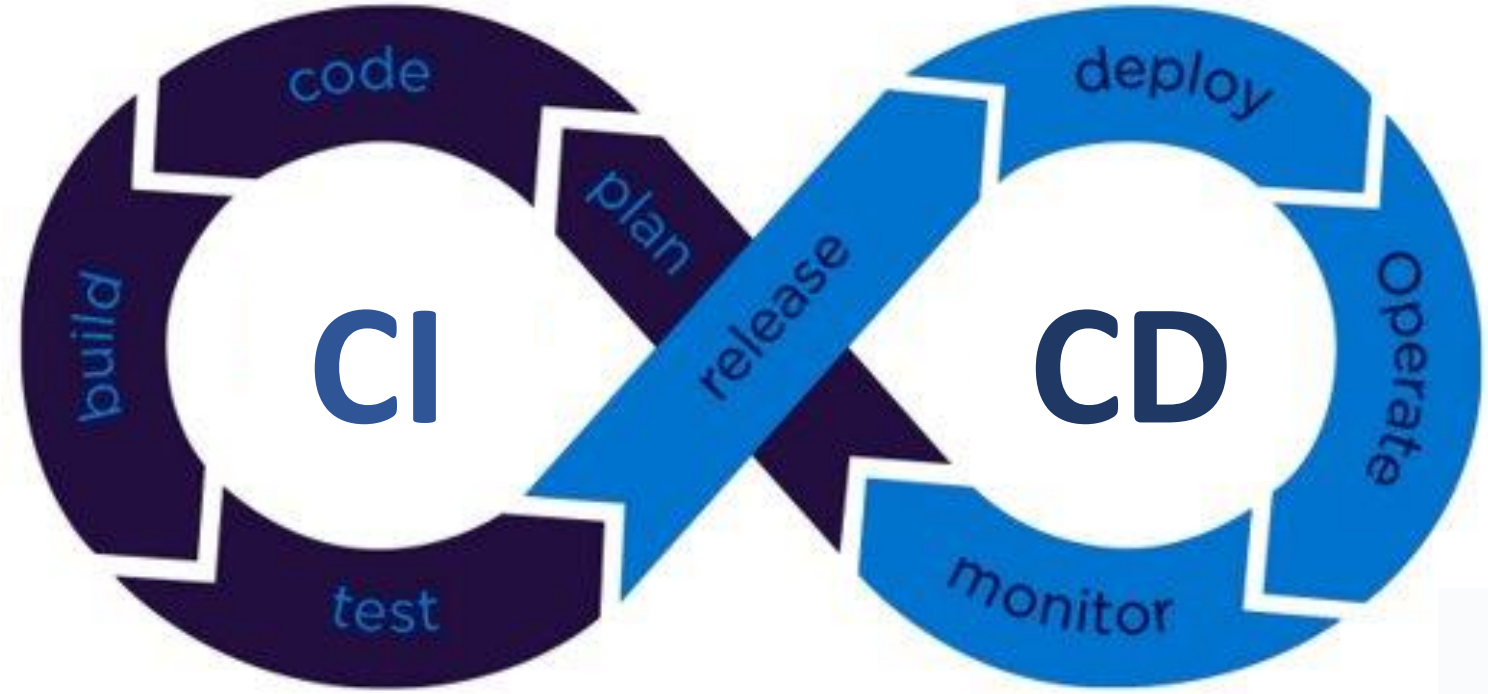
DevOps

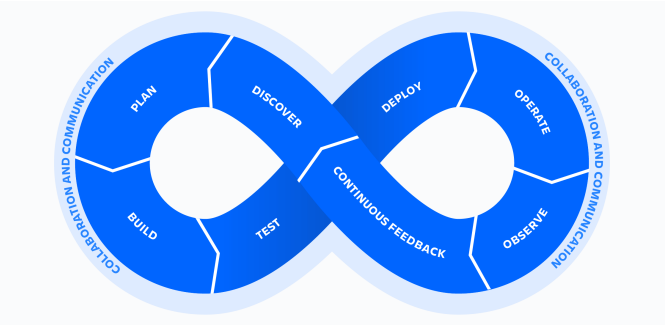

PLAN
DISCOVER
DEPLOY
OPERATE
BUILD
TEST
CONTINUOUS FEEDBACK
OBSERVE
COLLABORATION AND COMMUNICATION
COLLABORATION AND COMMUNICATION

CI/CD Pipeline

- A CI/CD pipeline **breaks down** into distinct subsets of **activities**. Typical pipeline stages include:
  - **Build**: The build phase of the application.
  - **Test**: The stage where the code is tested. Here automation can save time and effort.
  - **Release**: The stage where the application is pushed to the repository.
  - **Deployment**: in this phase the software is deployed in **production**.
  - **Validation and compliance**: the steps to validate a build required by the needs of the organization.
    - Software security scanning tools comparing it with known vulnerabilities (**CVE**).

CI/CD Pipeline





CI/CD Pipeline

- A CI/CD pipeline breaks down into distinct subsets of activities. Typical pipeline stages include:
- **Build**: The build phase of the application.
- **Test**: The stage where the code is tested.
  - **Unit tests**: elementary unit tests of the software
  - **Integration Testing**: Testing the interaction between the most basic software
  - **Functional Test**: test that given an input the Software (Backbox) provides the expected output
- **Release**: The stage where the application is pushed to the repository.
- **Deployment**: in this phase the software is distributed in staging.
  - **Test**: the phase in which the software is tested in execution.
  - **Deployment Testing**: Software installation testing, for example on different platforms
  - **Load/Stress Test**: Software load test
- **Validation and Compliance**:
  - **Acceptance test**: validation of requirements
  - **Security test**: comparison with known vulnerabilities (CVE).
- **Deployment**: the software is distributed in production
  - **Pre-production installation**
  - **Smoke test**
  - **Production installation**

CI/CD Pipeline

- A CI/CD pipeline breaks down into distinct subsets of activities. Typical pipeline stages include:
- **Build**: The build phase of the application.
- **Test**: The stage where the code is tested.
  - **Unit tests**: elementary unit tests of the software
  - **Integration Testing**: Testing the interaction between the most basic software
  - **Functional Test**: test that given an input the Software (Backbox) provides the expected output
- **Release**: The stage where the application is pushed to the repository.
- **Deployment**: in this phase the software is distributed in staging.
  - **Test**: the phase in which the software is tested in execution.
  - **Deployment Testing**: Software installation testing, for example on different platforms
  - **Load/Stress Test**: Software load test
- **Validation and Compliance**:
  - **Acceptance test**: validation of requirements
  - **Security test**: comparison with known vulnerabilities (CVE).
- **Deployment**: the software is distributed in production
  - **Pre-production installation**
  - **Smoke test**
  - **Production installation**

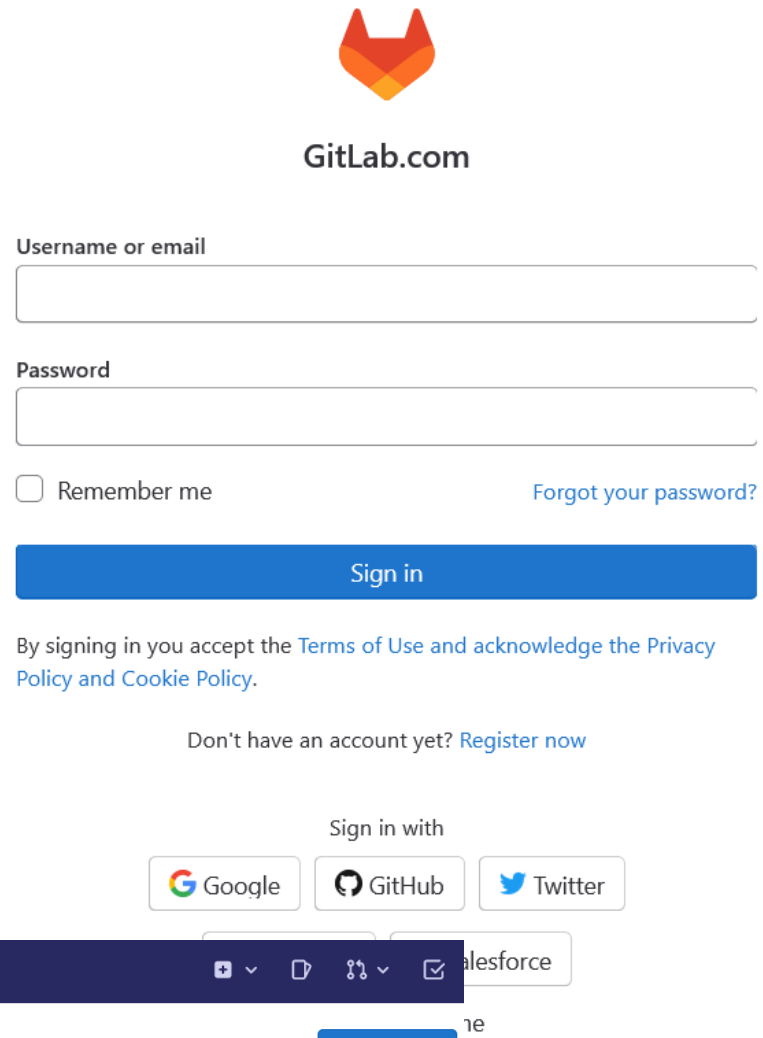
The situation can be even more complicated

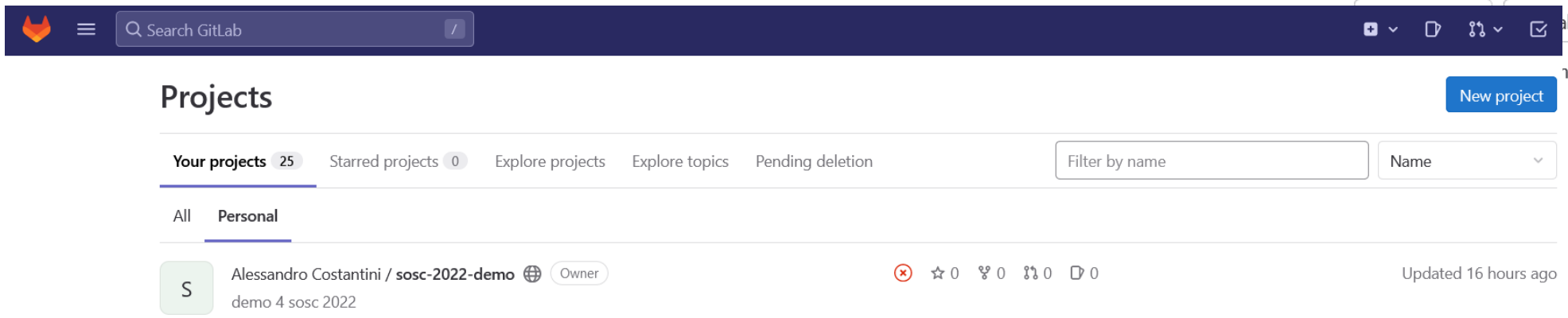
Tools

- **GitLab CI**
- **GitHub Actions**
- Jenkins
- Travis CI
- Bamboo/Bitbucket Pipelines (Atlassian)
- …

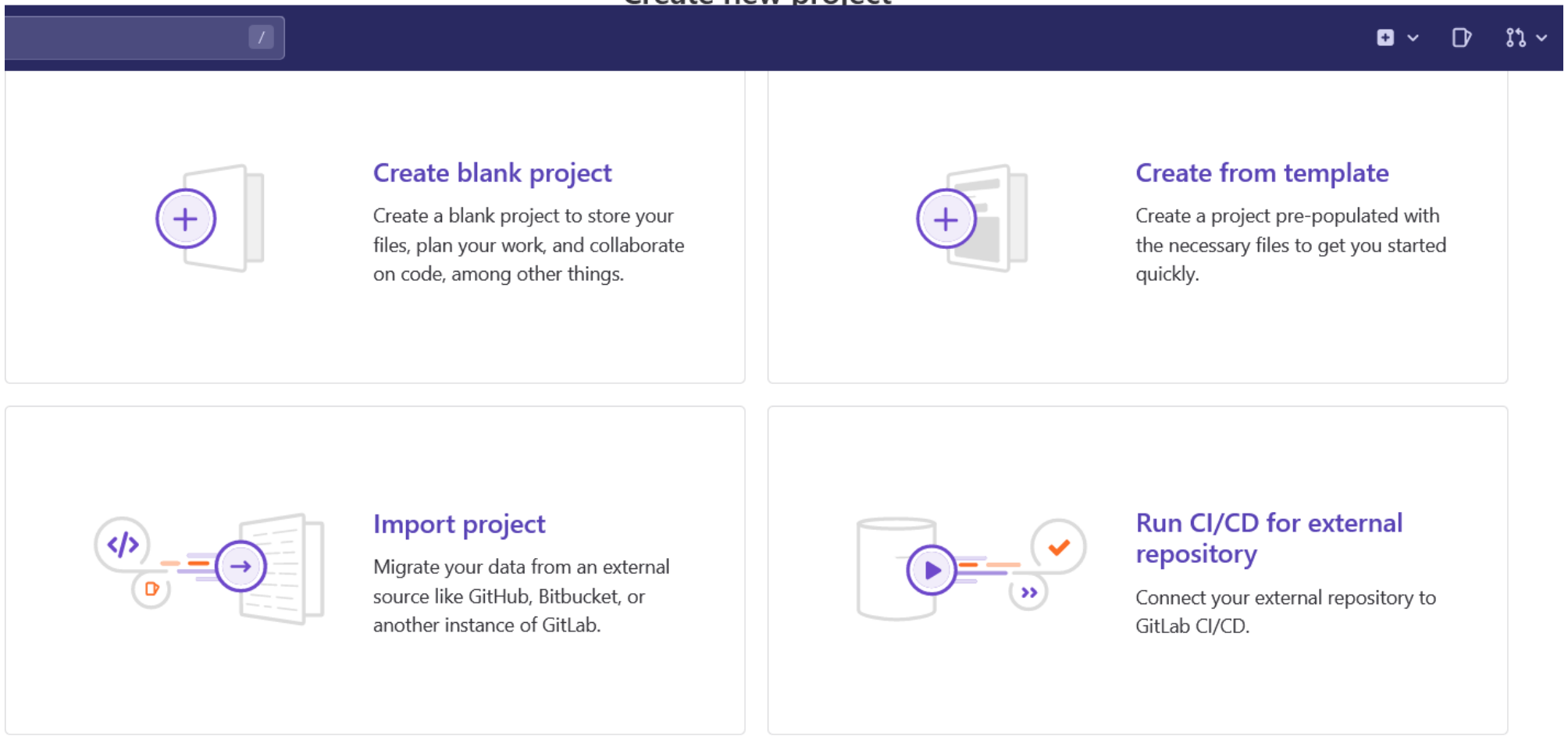
Working whit Git(Lab)

- https://gitlab.com/users/sign_in





Create new project



New project > Import project

Import project from

History

GitLab export | GitHub | Bitbucket Cloud | Bitbucket Server | FogBugz | Gitea | Repository by URL

Manifest file

- https://gitlab.com/alexcos78/sosc-2022-demo.git

Git repository URL

https://gitlab.com/alexcos78/sosc-2022-demo.git

Project URL

https://gitlab.com/ alexcos78 /

Project slug

sosc-2022-demo-2

Pick a group or namespace where you want to create this project.

Want to organize several dependent projects under the same namespace? Create a group.

Project description (optional)

Description format

Visibility Level

- Private
  Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.
- Public
  The project can be accessed without any authentication.

S

sosc-2022-demo-2

Project ID: 41411060

Star 0 | Fork 0

43 Commits | **2** Branches | **0** Tags | **133 KB** Project Storage

Update .gitlab-ci.yml
Alessandro Costantini authored 16 hours ago

4d8a263e

main | sosc-2022-demo-2 / | Find file | Web IDE | Clone | Gitpod

README | CI/CD configuration | Add LICENSE | Add CHANGELOG | Add CONTRIBUTING | Add Kubernetes cluster | Configure Integrations

Name	Last commit	Last update
.gitlab-ci.yml	Update .gitlab-ci.yml	16 hours ago
README.md	Create README.md	1 day ago

- https://gitlab.com/alexcos78/sosc-2022-demo

INFN
Istituto Nazionale di Fisica Nucleare

S

sosc-2022-demo
Project ID: 41397276

Star 0 | Fork 0

39 Commits | **2** Branches | **0** Tags | **348 KB** Project Storage

demo 4 sosc 2022

Update .gitlab-ci.yml
Alessandro Costantini authored 16 hours ago

main | sosc-2022-demo / | Find file | Web IDE

README | CI/CD configuration | Add LICENSE | Add CHANGELOG | Add CONTRIBUTING | Add K

Configure Integrations

Name	Last commit
.gitlab-ci.yml	Update .gitlab-ci.yml
README.md	Create README.md

.gitlab-ci.yml 415 bytes

```yaml
# .gitlab-ci.yml

# The names and order of the pipeline stages
stages:
  - build
  - test
  - deploy

build-job-example:
  stage: build
  script:
    - echo "Building the 'Hello World' app for " $GITLAB_USER_LOGIN

test-job-example:
  stage: test
  script:
    - echo "Testing the 'Hello World' app"

deploy-job-example:
  stage: deploy
  script:
    - echo "Deploying the 'Hello World' app in" $CI_COMMIT_BRANCH
```

Git has to be intalled in your platform

https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

```
alexc@LAPTOP-590KG1CS MINGW64 ~
$ git --version
git version 2.38.0.windows.1
```

https://www.atlassian.com/git/glossary

```
alexc@LAPTOP-590KG1CS MINGW64 ~/sosc-2022-demo-2 (main)
$ git clone https://gitlab.com/alexcos78/sosc-2022-demo-2.git
Cloning into 'sosc-2022-demo-2'...
remote: Enumerating objects: 168, done.
remote: Counting objects: 100% (168/168), done.
remote: Compressing objects: 100% (93/93), done.
remote: Total 168 (delta 37), reused 168 (delta 37), pack-reused 0
Receiving objects: 100% (168/168), 81.47 KiB | 6.27 MiB/s, done.
Resolving deltas: 100% (37/37), done.
```

https://www.atlassian.com/git/glossary

```
alexc@LAPTOP-590KG1CS MINGW64 ~
$ cd sosc-2022-demo-2/


alexc@LAPTOP-590KG1CS MINGW64 ~/sosc-2022-demo-2 (main)
$ ls -la
total 18
drwxr-xr-x 1 alexc 197609   0 Nov 28 10:38 ./
drwxr-xr-x 1 alexc 197609   0 Nov 28 10:38 ../
drwxr-xr-x 1 alexc 197609   0 Nov 28 10:38 .git/
-rw-r--r-- 1 alexc 197609 437 Nov 28 10:38 .gitlab-ci.yml
-rw-r--r-- 1 alexc 197609  36 Nov 28 10:38 README.md
```

```
alexc@LAPTOP-590KG1CS MINGW64 ~/sosc-2022-demo-2 (main)
$ git config --global user.email "alessandro.costantini@cnaf.infn.it"

alexc@LAPTOP-590KG1CS MINGW64 ~/sosc-2022-demo-2 (main)
$ git config --global user.name "Alessandro Costantini"
```

```
alexc@LAPTOP-590KG1CS MINGW64 ~/sosc-2022-demo-2 (main)
$ cat .git/config
…
[remote "origin"]
        url = https://gitlab.com/alexcos78/sosc-2022-demo-2.git
        fetch = +refs/heads/*:refs/remotes/origin/*
[branch "main"]
        remote = origin
        merge = refs/heads/main

alexc@LAPTOP-590KG1CS MINGW64 ~/sosc-2022-demo-2 (main)
$ git branch
* main
```

```
alexc@LAPTOP-590KG1CS MINGW64 ~/sosc-2022-demo-2 (main)
$ vim README.md


alexc@LAPTOP-590KG1CS MINGW64 ~/sosc-2022-demo-2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working
directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

```
alexc@LAPTOP-590KG1CS MINGW64 ~/sosc-2022-demo-2 (main)
$ git commit -a -m "Update readme"
[main ee48787] Update readme
 1 file changed, 1 insertion(+)

alexc@LAPTOP-590KG1CS MINGW64 ~/sosc-2022-demo-2 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

```
alexc@LAPTOP-590KG1CS MINGW64 ~/sosc-2022-demo-2 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 329 bytes | 329.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://gitlab.com/alexcos78/sosc-2022-demo-2.git
   4d8a263..ee48787  main -> main
```

S

sosc-2022-demo-2

Project ID: 41411060

Star 0 Fork 0

44 Commits **2** Branches **0** Tags **215 KB** Project Storage

Update readme

Alessandro Costantini authored 3 minutes ago

ee48787b

Commit ee48787b authored 3 minutes ago by **Alessandro Costantini**

Browse files | Options

Update readme

parent 4d8a263e main

No related merge requests found

Pipeline #706726474 failed

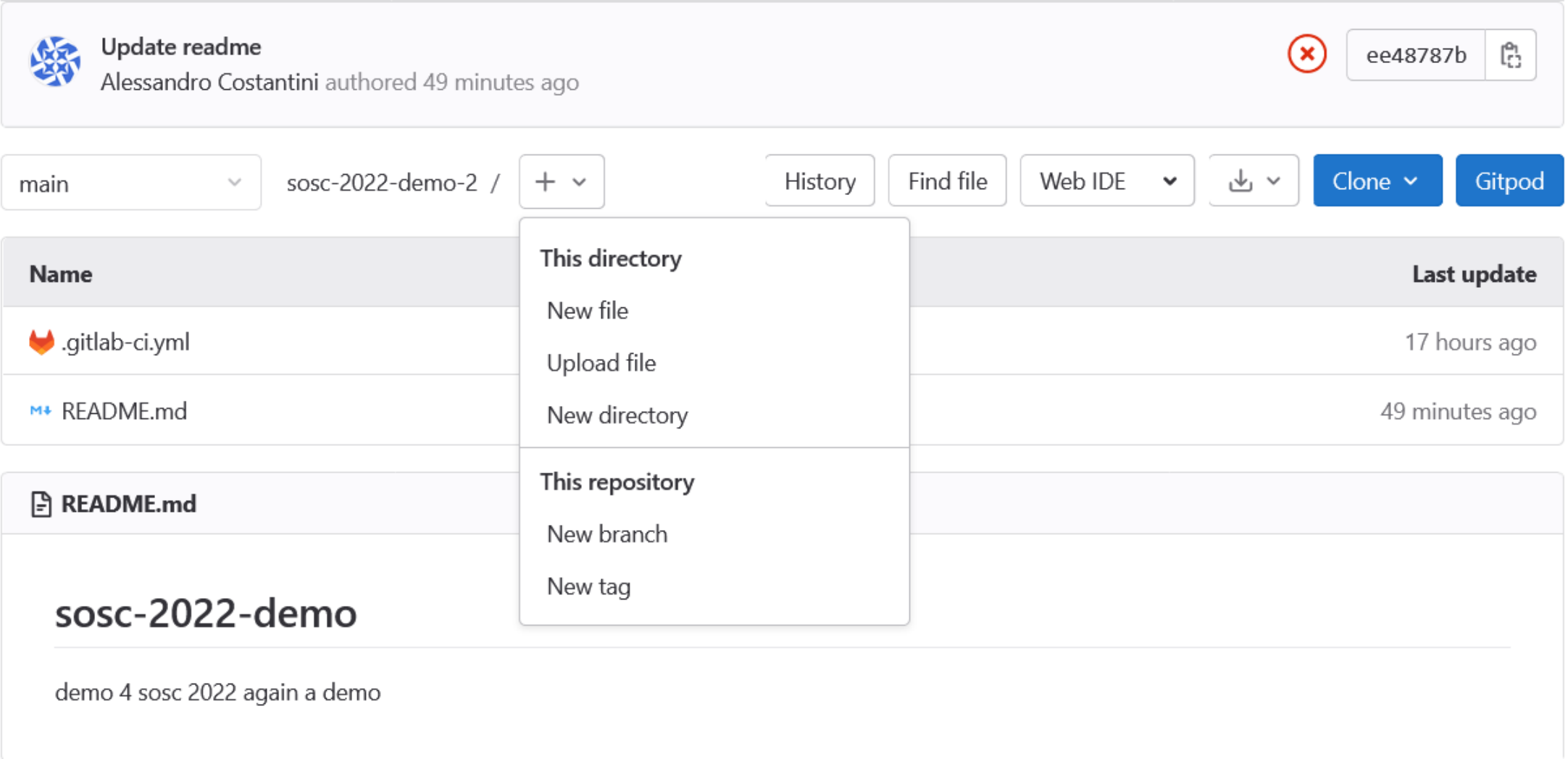
Changes 1 | Pipelines 1

Showing 1 changed file with 1 addition and 0 deletions

Hide whitespace changes | Inline | Side-by-side

README.md +1 -0 View file @ee48787b

```
1  1  # sosc-2022-demo
2  2  demo 4 sosc 2022
   3 + again a demo
```


Update readme
Alessandro Costantini authored 49 minutes ago
ee48787b
main
sosc-2022-demo-2 /
History
Find file
Web IDE
Clone
Gitpod
This directory
New file
Upload file
New directory
This repository
New branch
New tag
Name
Last update
.gitlab-ci.yml
17 hours ago
README.md
49 minutes ago
README.md
sosc-2022-demo
demo 4 sosc 2022 again a demo

New Branch

Branch name developer

Create from main

Existing branch name, tag, or commit SHA

Create branch Cancel

New Tag

Do you want to create a release with the new tag? You can do that in the New release page.

Tag name

v1.0.0

Create from

main

Existing branch name, tag, or commit SHA

Message

Fisrt release

Optionally, add a message to the tag. Leaving this blank creates a lightweight tag.

Create tag Cancel

```
alexc@LAPTOP-590KG1CS MINGW64 ~/sosc-2022-demo-2
(main)
$ git fetch
From https://gitlab.com/alexcos78/sosc-2022-demo-2
 * [new branch]      developer  -> origin/developer

alexc@LAPTOP-590KG1CS MINGW64 ~/sosc-2022-demo-2
(main)
$ git branch -r
  origin/HEAD -> origin/main
  origin/alexcos78-main-patch-00052
  origin/developer
  origin/main
```

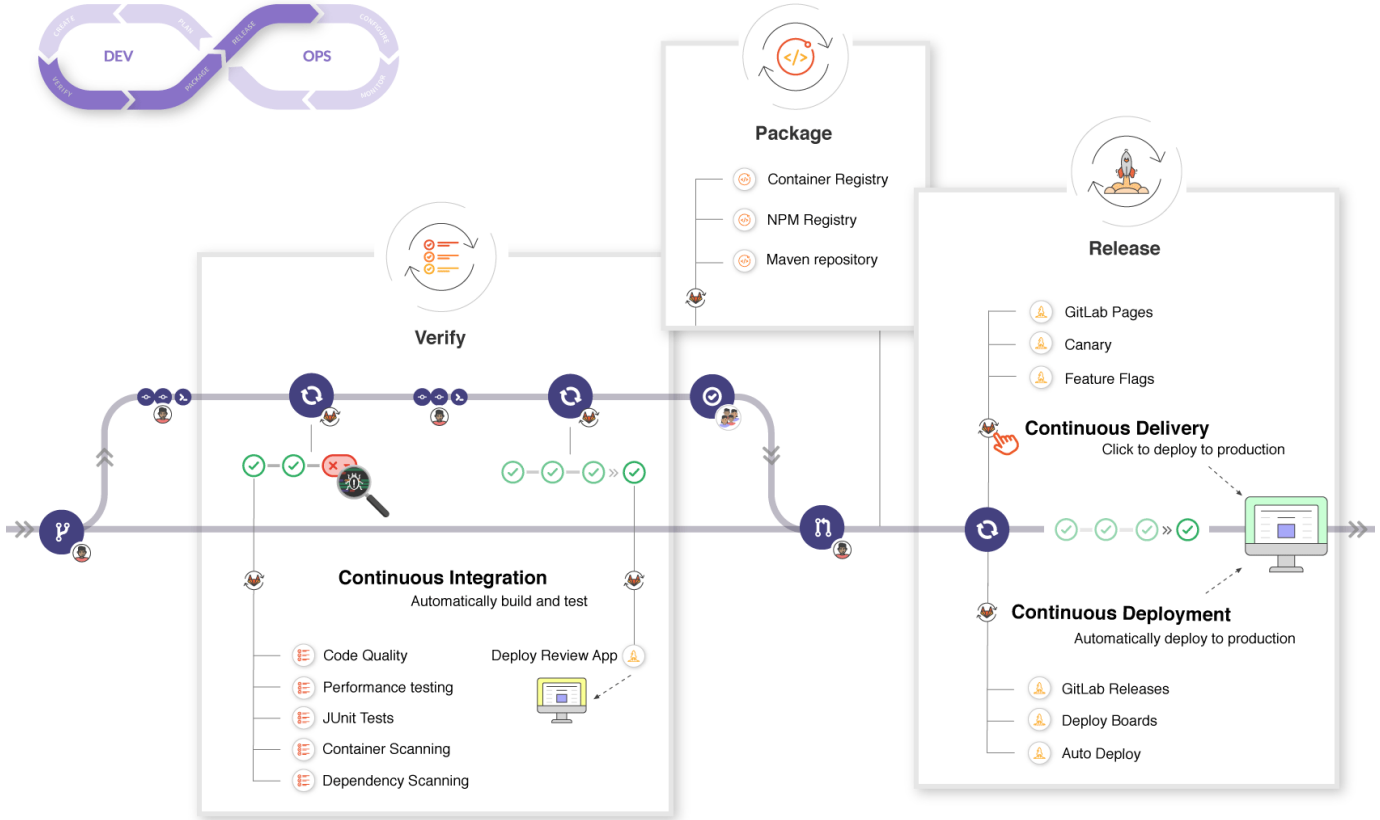
```
alexc@LAPTOP-590KG1CS MINGW64 ~/sosc-2022-demo-2
(main)
$ git checkout developer
Switched to a new branch 'developer'
branch 'developer' set up to track 'origin/developer'.

alexc@LAPTOP-590KG1CS MINGW64 ~/sosc-2022-demo-2
(developer)
$ git branch
* developer
  main
```

CI/CD in GitLab

CI/CD in GitLab

- The **CI/CD is integrated in GitLab** and allows software development according to the methodologies
- **Continuous Integration**
  - A developer has his repository in GitLab and with each "push" on the repository, possibly also for development branches, a series of scripts starts that compile and test the application
- **Continuous Delivery**
  - A step further than CI; the application is released for the repository
- **Continuous Deployment**
  - The application released is put into production automatically without manual intervention



https://docs.gitlab.com/ee/ci/quick_start/index.html

CI/CD in GitLab

What is needed? A repository and a file.

The CI/CD is configured through a **.gitlab-ci.yml** file present in the root of the repository.

When pushed to the repository, the file executes a pipeline: a set of instructions that execute jobs on a runner

The file .gitlab-ci.yml is a YAML file

- https://en.wikipedia.org/wiki/YAML
- https://yaml.org/

```
### Test simple pipeline

simple:
  script:
    - echo "bellashell"
```

In this file, you define:

- The structure and order of jobs that the runner should execute.
- The decisions the runner should make when specific conditions are encountered.

CI/CD in GitLab

Pipeline: is the highest level component of the CI/CD

- Simple pipeline
- Complex pipeline
  - https://gitlab.com/gitlab-org/gitlab/blob/master/.gitlab-ci.yml

```yaml
### Test simple pipeline

simple:
  script:
    - echo "bellashell"
```

```
14 Skipping Git submodules setup
15 Executing "step_script" stage of the job script
16 Using docker image sha256:300e315adb2f96afe5f0b2
   b8c04efc1 ...
17 $ echo "bellashell"
18 bellashell
19 Cleaning up file based variables
20 Job succeeded
```

```yaml
stages:
  - sync
  - prepare
  - build-images
  - fixtures
  - test
  - post-test
  - review-prepare
  - review
  - dast
  - qa
  - post-qa
  - pages
  - notify
```

```yaml
variables:
  RAILS_ENV: "test"
  NODE_ENV: "test"
  # we override the max_old_space_size to prevent OOM errors
  NODE_OPTIONS: --max_old_space_size=3584
  SIMPLECOV: "true"
  GIT_DEPTH: "20"
  GIT_SUBMODULE_STRATEGY: "none"
  GET_SOURCES_ATTEMPTS: "3"
  KNAPSACK_RSPEC_SUITE_REPORT_PATH: knapsack/report-master.json
  FLAKY_RSPEC_SUITE_REPORT_PATH: rspec_flaky/report-suite.json
  RSPEC_TESTS_MAPPING_PATH: crystalball/mapping.json
  RSPEC_PACKED_TESTS_MAPPING_PATH: crystalball/packed-mapping.json
  BUILD_ASSETS_IMAGE: "false"
  ES_JAVA_OPTS: "-Xms256m -Xmx256m"
  ELASTIC_URL: "http://elastic:changeme@elasticsearch:9200"
  DOCKER_VERSION: "20.10.1"
  CACHE_CLASSES: "true"
```

```yaml
include:
  - local: .gitlab/ci/build-images.gitlab-ci.yml
  - local: .gitlab/ci/cache-repo.gitlab-ci.yml
  - local: .gitlab/ci/cng.gitlab-ci.yml
  - local: .gitlab/ci/docs.gitlab-ci.yml
  - local: .gitlab/ci/frontend.gitlab-ci.yml
  - local: .gitlab/ci/global.gitlab-ci.yml
  - local: .gitlab/ci/memory.gitlab-ci.yml
  - local: .gitlab/ci/pages.gitlab-ci.yml
  - local: .gitlab/ci/qa.gitlab-ci.yml
  - local: .gitlab/ci/reports.gitlab-ci.yml
  - local: .gitlab/ci/rails.gitlab-ci.yml
  - local: .gitlab/ci/vendored-gems.gitlab-ci.yml
  - local: .gitlab/ci/review.gitlab-ci.yml
  - local: .gitlab/ci/rules.gitlab-ci.yml
  - local: .gitlab/ci/setup.gitlab-ci.yml
  - local: .gitlab/ci/dev-fixtures.gitlab-ci.yml
  - local: .gitlab/ci/test-metadata.gitlab-ci.yml
  - local: .gitlab/ci/yaml.gitlab-ci.yml
  - local: .gitlab/ci/releases.gitlab-ci.yml
  - local: .gitlab/ci/notify.gitlab-ci.yml
  - local: .gitlab/ci/dast.gitlab-ci.yml
  - local: .gitlab/ci/workhorse.gitlab-ci.yml
  - local: .gitlab/ci/graphql.gitlab-ci.yml
```

CI/CD in GitLab

- This example shows three jobs:
  - build-job-example, test-job-example, t deploy-job-example.
- The comments listed in the echo commands are displayed in the UI when you view the jobs.
- The values for the predefined variables $GITLAB_USER_LOGIN and $CI_COMMIT_BRANCH are populated when the jobs run.

.gitlab-ci.yml 415 bytes

```yaml
# .gitlab-ci.yml

# The names and order of the pipeline stages
stages:
  - build
  - test
  - deploy

build-job-example:
  stage: build
  script:
    - echo "Building the 'Hello World' app for " $GITLAB_USER_LOGIN

test-job-example:
  stage: test
  script:
    - echo "Testing the 'Hello World' app"

deploy-job-example:
  stage: deploy
  script:
    - echo "Deploying the 'Hello World' app in" $CI_COMMIT_BRANCH

```

CI/CD in GitLab

Keywords stages & stage

Stages

- Define the *stage* containig groups of *job*
- The order of the arguments defines the execution order of the *job*
- Is globally defined
- If not defined, the default stage are build, test, deploy

```
stages:
  - build
  - test
  - deploy
```

Stage

- Defines the *job* executed in the stage
- If *stages* is not defined, there are 5 default stage (executed in the order) *.pre, build, test, deploy, .post*
- To a *job* without *stage* is assingned the stage *test*

```
stages:
  - build
  - test
  - deploy

job 0:
  stage: .pre
  script: make something useful before build stage

job 1:
  stage: build
  script: make build dependencies

job 2:
  stage: build
  script: make build artifacts

job 3:
  stage: test
  script: make test

job 4:
  stage: deploy
  script: make deploy
```

CI/CD in GitLab

Pipeline

Besides the keywords to be used to "build" the pipeline, there are environmental variables that are defined in the execution of the jobs and are useful for

- control the behavior of jobs and pipeline
- Assume a value to be used in the *job*
- avoid *hard-coded* values in the file .gitlab-ci.yml
- https://docs.gitlab.com/ee/ci/variables/predefined_variables.html
- define *custom variables*

```
test_variable:
  stage: test
  script:
    - echo $CI_JOB_STAGE
```

```
variables:
  TEST_VAR: "All jobs can use this variable's value"

job1:
  variables:
    TEST_VAR_JOB: "Only job1 can use this variable's value"
  script:
    - echo $TEST_VAR and $TEST_VAR_JOB
```

- *pipeline* generating an *artifact* (a product of the job)

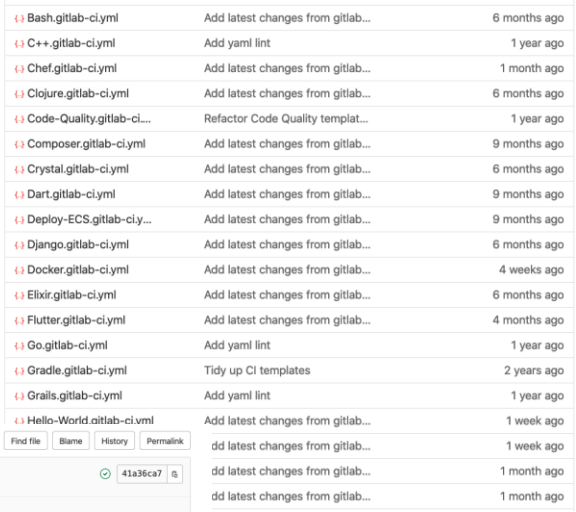
The *artifact* is a pdf that will be removed after a week *expire_in* and it is in the folder *paths* related to the repository where the job is executed

```
pdf:
  script: xelatex mycv.tex
  artifacts:
    paths:
      - mycv.pdf
    expire_in: 1 week
```

CI/CD in GitLab

There are many temlates available in the GUI for .gitlab-ci.yml

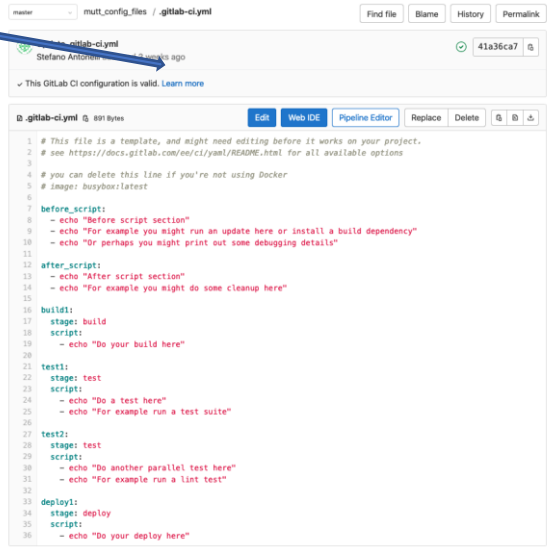
https://gitlab.com/gitlab-org/gitlab-foss/tree/master/lib/gitlab/ci/templates



Bash.gitlab-ci.yml	Add latest changes from gitlab...	6 months ago
C++.gitlab-ci.yml	Add yaml lint	1 year ago
Chef.gitlab-ci.yml	Add latest changes from gitlab...	1 month ago
Clojure.gitlab-ci.yml	Add latest changes from gitlab...	6 months ago
Code-Quality.gitlab-ci....	Refactor Code Quality templat...	1 year ago
Composer.gitlab-ci.yml	Add latest changes from gitlab...	9 months ago
Crystal.gitlab-ci.yml	Add latest changes from gitlab...	6 months ago
Dart.gitlab-ci.yml	Add latest changes from gitlab...	9 months ago
Deploy-ECS.gitlab-ci.y...	Add latest changes from gitlab...	9 months ago
Django.gitlab-ci.yml	Add latest changes from gitlab...	6 months ago
Docker.gitlab-ci.yml	Add latest changes from gitlab...	4 weeks ago
Elixir.gitlab-ci.yml	Add latest changes from gitlab...	6 months ago
Flutter.gitlab-ci.yml	Add latest changes from gitlab...	4 months ago
Go.gitlab-ci.yml	Add yaml lint	1 year ago
Gradle.gitlab-ci.yml	Tidy up CI templates	2 years ago
Grails.gitlab-ci.yml	Add yaml lint	1 year ago
Hello-World.gitlab-ci.yml	Add latest changes from gitlab...	1 week ago
	dd latest changes from gitlab...	1 week ago
	dd latest changes from gitlab...	1 month ago
	dd latest changes from gitlab...	1 month ago

From the GitLab editor, file can be modified

https://docs.gitlab.com/ee/ci/yaml/README.html#variables

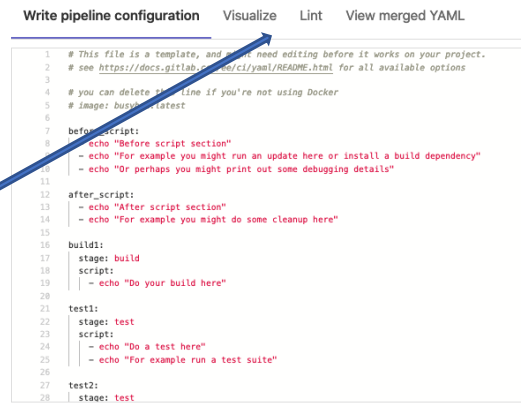


Here you can:

Edit the *pipeline*

Visualize the *pipeline*

Verify the sintax



CI/CD in GitLab

Who is executing the conde in *.gitlab-ci.yml*?

- the *runner*
- In GitLab *runners* can be used by presenting the credit card
- In GitLab INFN there are *shared runners*, 16 *runner* available for the users

https://docs.gitlab.com/ee/ci/quick_start/index.html#ensure-you-have-runners-available

#164 (a5c4f63a)
BALTIG-RUNNER-WIN-1
visual studio | windows

#147 (a4b08b59)
baltig-runner-10.cnaf.infn.it
docker | runner-10 | shared

#262 (iGBpFA2h)
baltig-runner-macos-02.cnaf.infn.it
macos | mojave | osx | shared

#263 (3jYTVdNv)
baltig-runner-windows-02.cnaf.infn.it
shared | visual studio | windows

#146 (1e8a9978)
batilg-runner-9.cnaf.infn.it
docker | runner-09 | shared

#261 (KopTsMzi)
baltig-runner-macos-01.cnaf.infn.it
macos | mojave | osx | shared

#149 (72dcb49d)
baltig-runner-12.cnaf.infn.it
docker | runner-12 | shared

#153 (eab6c56d)
baltig-runner-16.cnaf.infn.it
docker | runner-16 | shared

The *job is executed by one runner*

In the *dashboard* can be seen the execution time of each *job*

```
1  Running with gitlab-runner 13.11.0 (7f7a4bb0)
2    on baltig-runner-8 3242bd47
3  Preparing the "docker" executor

4  Using Docker executor with image centos:latest ...
5  Pulling docker image centos:latest ...
6  Using docker image sha256:300e315adb2f96afe5f0b2780b87f28ae95231fe3bdd1e16b9ba606307728f55 for centos:latest
   b8c04efc1 ...
8  Preparing environment

9  Running on runner-3242bd47-project-3564-concurrent-0 via baltig-runner-8.cnaf.infn.it...
11 Getting source from Git repository
```

CI/CD in GitLab

Dashboard enable the control of CI (pipeline, job...)

failed	#45938	master a274d9eb Update .gitlab-ci.y...
failed	#45931 yaml invalid error	master ec871f1d Update .gitlab-ci.y...

Merge requests 29
CI/CD
Pipelines
Editor
Jobs
Schedules

Status	Pipeline	Triggerer	Commit	Stages	Duration
passed	#45968		master 6f94a661 Update index.html		00:01:08 4 days ago
passed	#45967		master ab032976 Update .gitlab-ci.y...		00:0 4 day
passed	#45966		master f42480a3 Update .gitlab-ci.y...		00:08:25 4 days ago

Download pages:archive artifact
Download build:archive artifact

Repository
Issues 29
Merge requests 29
CI/CD
Pipelines
Editor
Jobs

Status	Job	Pipeline	Stage	Name	Duration	Coverage
passed	#165441 master 6f94a661	#45968 by	pages	pages	00:00:36 4 days ago	
passed	#165440 master 6f94a661	#45968 by	build	build	00:00:31 4 days ago	
passed	#165438 master ab032976	#45967 by	pages	pages	00:00:30 4 days ago	
passed	#165437 master ab032976	#45967 by	build	build	00:06:30 4 days ago	
passed	#165435 master f42480a3	#45966 by	pages	pages	00:00:37 4 days ago	

Jobs
Schedules
Security & Compliance
Operations
Packages & Registries
Analytics
Wiki
Snippets
Members

```
38 public/:
39 total 4
40 -rw-rw-rw- 1 root root 184 Apr 23 21:42 index.html
41 $ cp pdf/slide_ci_artifact.pdf public/
42 $ ls -ltr pdf/ public
43 pdf/:
44 total 240
45 -rw-r--r-- 1 root root 242302 Apr 23 21:41 slide_ci_artifact.pdf
46 public:
47 total 244
48 -rw-rw-rw- 1 root root    184 Apr 23 21:42 index.html
49 -rw-r--r-- 1 root root 242302 Apr 23 21:42 slide_ci_artifact.pdf
51 Uploading artifacts for successful job
52 Uploading artifacts...
```

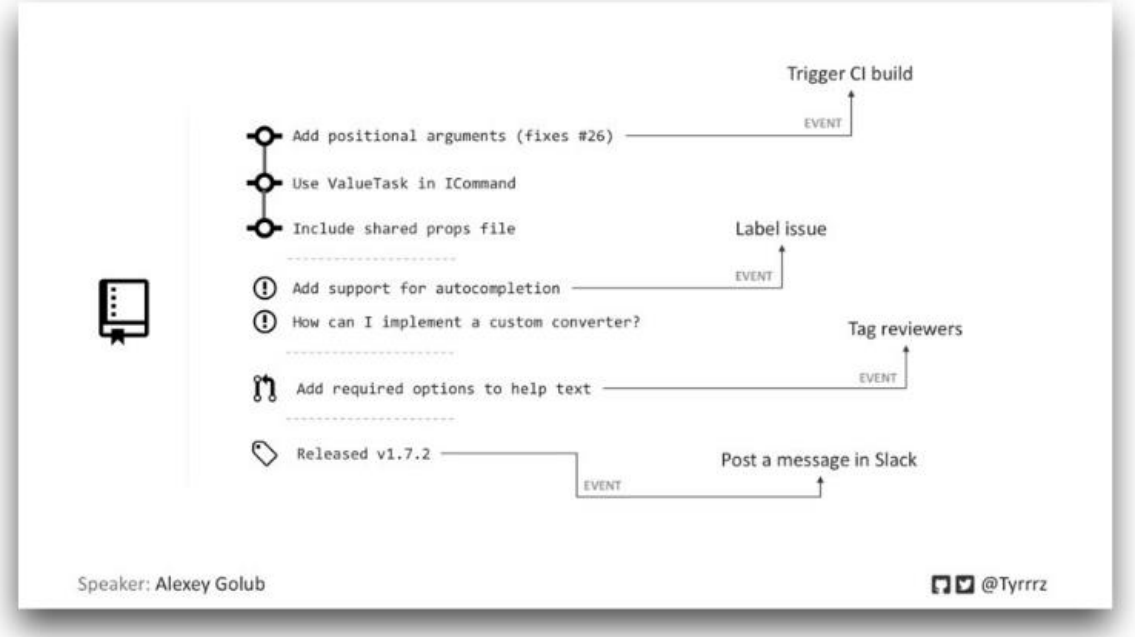
GitHub Actions

GitHub Actions

- https://docs.github.com/en/actions
- Available since Nov. 13, 2019
- Implemented on Microsoft Azure Pipelines
- Tightly integrated with the GitHub API
- YAML-based configuration
- Modular architecture, community-driven
- Windows, Linux, MacOS, self hosted runners
- Free for public repositories

More than CI/CD?

- GH Actions provides tools to automate any task on your Github-hosted repository



An example

main sosc-2022-demo / .github / workflows / simple_build.yml

```yaml
name: Simple workflow

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Write a multi-line message
      run: |
       echo This demo file shows a
       echo very basic and easy-to-understand workflow.
```

Name of the workflow

https://github.com/alexcos78/sosc-2022-demo/blob/main/.github/workflows/simple_build.yml

main sosc-2022-demo / .github / workflows / simple_build.yml

```yaml
name: Simple workflow

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Write a multi-line message
      run: |
       echo This demo file shows a
       echo very basic and easy-to-understand workflow.
```

Events that trigger the workflow

https://github.com/alexcos78/sosc-2022-demo/blob/main/.github/workflows/simple_build.yml

main sosc-2022-demo / .github / workflows / simple_build.yml

```yaml
name: Simple workflow

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Write a multi-line message
      run: |
       echo This demo file shows a
       echo very basic and easy-to-understand workflow.
```

Workflow jobs

We have only one job in this workflow, the **build** job

Jobs

A job is a set of *steps* that execute in the same runner. By default, a workflow with multiple jobs will run those jobs in parallel. You can also configure a workflow to run job sequentially. For example, a workflow can have two sequential jobs that buold and test code, where the test job is dependent on the status of the buold job. If the build job fails, the test job will not run.

https://github.com/alexcos78/sosc-2022-demo/blob/main/.github/workflows/simple_build.yml

main sosc-2022-demo / .github / workflows / simple_build.yml

```yaml
name: Simple workflow

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Write a multi-line message
      run: |
        echo This demo file shows a
        echo very basic and easy-to-understand workflow.
```

The job runs on an ubuntu runner

Supported runners and hardware resources

Hardware specification for Windows and Linux virtual machines:

- 2-core CPU
- 7 GB of RAM memory
- 14 GB of SSD disk space

Hardware specification for macOS virtual machines:

- 3-core CPU
- 14 GB of RAM memory
- 14 GB of SSD disk space

Virtual environment	YAML workflow label
Windows Server 2019	`windows-latest` or `windows-2019`
Windows Server 2016	`windows-2016`
Ubuntu 20.04	`ubuntu-latest` or `ubuntu-20.04`
Ubuntu 18.04	`ubuntu-18.04`
Ubuntu 16.04	`ubuntu-16.04`

https://docs.github.com/en/actions/using-github-hosted-runners/about-github-hosted-runners

https://github.com/alexcos78/sosc-2022-demo/blob/main/.github/workflows/simple_build.yml

main sosc-2022-demo / .github / workflows / simple_build.yml

```yaml
name: Simple workflow

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Write a multi-line message
      run: |
       echo This demo file shows a
       echo very basic and easy-to-understand workflow.
```

The steps that make up this build job

Steps

A step is an individual task that can run commands in a job.
A step can be either an action or a shell command.
Each step in a job executes on the same runner, allowing actions in that job to share data with each other.

https://github.com/alexcos78/sosc-2022-demo/blob/main/.github/workflows/simple_build.yml

main sosc-2022-demo / .github / workflows / simple_build.yml

```yaml
name: Simple workflow

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Write a multi-line message
      run: |
       echo This demo file shows a
       echo very basic and easy-to-understand workflow.
```

The first step is the execution of the checkout action

Actions

Actions are standalone commands that are combined into steps to crate a job.
Actions are the smallest portable building block of a workflow.
You can create your own actions, or use actions created by the CitHub community.
To use an action in a workflow, you must include it as a step

https://github.com/alexcos78/sosc-2022-demo/blob/main/.github/workflows/simple_build.yml

main sosc-2022-demo / .github / workflows / simple_build.yml

```yaml
name: Simple workflow

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Write a multi-line message
      run: |
       echo This demo file shows a
       echo very basic and easy-to-understand workflow.
```

The **run** keyword tells the job to execute a command on the runner.

https://github.com/alexcos78/sosc-2022-demo/blob/main/.github/workflows/simple_build.yml

main sosc-2022-demo / .github / workflows / simple_build.yml

```yaml
name: Simple workflow

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Write a multi-line message
      run: |
       echo This demo file shows a
       echo very basic and easy-to-understand workflow.
```

build

succeeded 13 minutes ago in 2s

- Set up job
- Run actions/checkout@v2
- Write a multi-line message
- Post Run actions/checkout@v2
- Complete job

main sosc-2022-demo / .github / workflows / simple_build.yml

```yaml
name: Simple workflow

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Write a multi-line message
      run: |
       echo This demo file shows a
       echo very basic and easy-to-understand workflow.
```

Set up job

```
Current runner version: '2.299.1'
▶ Operating System
▶ Runner Image
▶ Runner Image Provisioner
▶ GITHUB_TOKEN Permissions
Secret source: Actions
Prepare workflow directory
Prepare all required actions
Getting action download info
Download action repository 'actions/checkout@v2'
```

main sosc-2022-demo / .github / workflows / simple_build.yml

```yaml
name: Simple workflow

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Write a multi-line message
      run: |
        echo This demo file shows a
        echo very basic and easy-to-understand workflow.
```

Run actions/checkout@v2

```
1  ▶ Run actions/checkout@v2
12 Syncing repository: alexcos78/sosc-2022-demo
13 ▶ Getting Git version info
17 Temporarily overriding HOME='/home/runner/work/_temp/8100e56d-5236-4c08-929a
18 Adding repository directory to the temporary git global config as a safe dir
19 /usr/bin/git config --global --add safe.directory /home/runner/work/sosc-202
20 Deleting the contents of '/home/runner/work/sosc-2022-demo/sosc-2022-demo'
21 ▶ Initializing the repository
35 ▶ Disabling automatic garbage collection
37 ▶ Setting up auth
43 ▶ Fetching the repository
65 ▶ Determining the checkout info
66 ▶ Checking out the ref
70 /usr/bin/git log -1 --format='%H'
71 '48242f7866706c5223a419b8baee9bfb2d5a355f'
```

https://github.com/actions
https://github.com/marketplace?type=actions

main sosc-2022-demo / .github / workflows / simple_build.yml

```yaml
name: Simple workflow

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Write a multi-line message
      run: |
       echo This demo file shows a
       echo very basic and easy-to-understand workflow.
```

Write a multi-line message

```
1  ▶ Run echo This demo file shows a
5  This demo file shows a
6  very basic and easy-to-understand workflow.
```

main sosc-2022-demo / .github / workflows / simple_build.yml

```yaml
name: Simple workflow

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Write a multi-line message
      run: |
        echo This demo file shows a
        echo very basic and easy-to-understand workflow.
```

Post Run actions/checkout@v2

```
Post job cleanup.
/usr/bin/git version
git version 2.38.1
Temporarily overriding HOME='/home/runner/work/_temp/d4d52a78-3631-4e18-bf33-d4
Adding repository directory to the temporary git global config as a safe direct
/usr/bin/git config --global --add safe.directory /home/runner/work/sosc-2022-d
/usr/bin/git config --local --name-only --get-regexp core\.sshCommand
/usr/bin/git submodule foreach --recursive git config --local --name-only --get
/usr/bin/git config --local --name-only --get-regexp http\.https\:\/\/github\.c
http.https://github.com/.extraheader
/usr/bin/git config --local --unset-all http.https://github.com/.extraheader
/usr/bin/git submodule foreach --recursive git config --local --name-only --get
|| :
```

Complete job

```
Cleaning up orphan processes
```

An (bit more complex) example

```yaml
name: Python package

on: [push]

jobs:
  build:

    runs-on: ubuntu-latest
    strategy:
      fail-fast: false
      matrix:
        #python-version: ["3.8"]
        python-version: ["3.8", "3.9", "3.10"]


    steps:
    - uses: actions/checkout@v3
    - name: Set up Python ${{ matrix.python-version }}
      uses: actions/setup-python@v3
      with:
        python-version: ${{ matrix.python-version }}
    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        python -m pip install flake8 pytest
        if [ -f requirements.txt ]; then pip install -r requirements.txt; fi
    - name: Lint with flake8
      run: |
        # stop the build if there are Python syntax errors or undefined names
        flake8 . --count --select=E9,F63,F7,F82 --show-source --statistics
        # exit-zero treats all errors as warnings. The GitHub editor is 127 chars wide
        flake8 . --count --exit-zero --max-complexity=10 --max-line-length=127 --statistics

```

This workflow will install Python dependencies, run lint with a variety of Python versions

https://github.com/alexcos78/sosc-2022-demo/blob/main/.github/workflows/python-workflow.yml

An (bit more complex) example

```yaml
name: Python package

on: [push]

jobs:
  build:

    runs-on: ubuntu-latest
    strategy:
      fail-fast: false
      matrix:
        #python-version: ["3.8"]
        python-version: ["3.8", "3.9", "3.10"]


    steps:
    - uses: actions/checkout@v3
    - name: Set up Python ${{ matrix.python-version }}
      uses: actions/setup-python@v3
      with:
        python-version: ${{ matrix.python-version }}
    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        python -m pip install flake8 pytest
        if [ -f requirements.txt ]; then pip install -r requirements.txt; fi
    - name: Lint with flake8
      run: |
        # stop the build if there are Python syntax errors or undefined names
        flake8 . --count --select=E9,F63,F7,F82 --show-source --statistics
        # exit-zero treats all errors as warnings. The GitHub editor is 127 chars wide
        flake8 . --count --exit-zero --max-complexity=10 --max-line-length=127 --statistics

```

This workflow will install Python dependencies, run lint with a variety of Python versions

We provide input parameters to the action using the **with** keyword

https://github.com/alexcos78/sosc-2022-demo/blob/main/.github/workflows/python-workflow.yml

Istituto Nazionale di Fisica Nucleare

Search or jump to... / Pull requests Issues Codespaces Marketplace Explore

alexcos78 / sosc-2022-demo Public

Pin Unwatch 1

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Actions

New workflow

All workflows

Python package

Simple workflow

Management

Caches

Workflow name

All workflows

Showing runs from all workflows

Filter workflow runs

Tell us how to make GitHub Actions work better for you with three quick questions.

Give feedback

45 workflow runs

Event Status Branch Actor

Rename python-test.yml to python-workflow.yml
Simple workflow #15: Commit b4afe73 pushed by alexcos78
main
7 minutes ago
9s

Rename python-test.yml to python-workflow.yml
Python package #1: Commit b4afe73 pushed by alexcos78
main
7 minutes ago
18s

Update python-test.yml
Simple workflow #14: Commit 4b5192c pushed by alexcos78
main
7 minutes ago
12s

Istituto Nazionale di Fisica Nucleare

Search or jump to... / Pull requests Issues Codespaces Marketplace Explore

alexcos78 / sosc-2022-demo Public

Pin Unwatch 1

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Actions

New workflow

All workflows

Python package

Simple workflow

Management

Caches

Successful build

All workflows

Showing runs from all workflows

Filter workflow runs

Tell us how to make GitHub Actions work better for you with three quick questions.

Give feedback

45 workflow runs

Event Status Branch Actor

Rename python-test.yml to python-workflow.yml
Simple workflow #15: Commit b4afe73 pushed by alexcos78
main
7 minutes ago
9s

Rename python-test.yml to python-workflow.yml
Python package #1: Commit b4afe73 pushed by alexcos78
main
7 minutes ago
18s

Update python-test.yml
Simple workflow #14: Commit 4b5192c pushed by alexcos78
main
7 minutes ago
12s

Istituto Nazionale di Fisica Nucleare

Search or jump to... / Pull requests Issues Codespaces Marketplace Explore

alexcos78 / sosc-2022-demo Public

Pin Unwatch 1

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Actions

New workflow

All workflows

Python package

Simple workflow

Management

Caches

All workflows

Showing runs from all workflows

Filter workflow runs

Tell us how to make GitHub Actions work better for you with three quick questions.

Give feedback

Event filter

45 workflow runs

Event Status Branch Actor

Rename python-test.yml to python-workflow.yml
Simple workflow #15: Commit b4afe73 pushed by alexcos78
main
7 minutes ago
9s

Rename python-test.yml to python-workflow.yml
Python package #1: Commit b4afe73 pushed by alexcos78
main
7 minutes ago
18s

Update python-test.yml
Simple workflow #14: Commit 4b5192c pushed by alexcos78
main
7 minutes ago
12s

Search or jump to... / Pull requests Issues Codespaces Marketplace Explore

alexcos78 / sosc-2022-demo Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

alexcos78 Rename python-test.yml to python-workflow.yml b4afe73 11 minutes ago 38 commits

.github/wo 11 minutes ago

README.r 8 hours ago

All checks have passed
4 successful checks

- Python package / build (3.8) (push) Successful in 10s Details
- Simple workflow / build (push) Successful in 2s Details
- Python package / build (3.9) (push) Successful in 10s Details
- Python package / build (3.10) (push) Successful in 10s Details

Successful build for this commit

README.md

SOSC-

demo 4 sosc 2022

About

demo 4 sosc 2022

Readme
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Triggers

```yaml
# Trigger on push events on s
pecific branches
on:
  push:
    branches:
      - 'master'
      - 'release/*'
```

```yaml
# Trigger on manual dispatch
on: repository_dispatch
```

```yaml
# Trigger every midnight UTC
on:
  schedule:
    - cron: '0 0 * * *'
```

```yaml
# Trigger when an issue is opened o
r labeled
on:
  issues:
    types: [opened, labeled]
```

Argo

What is Argo?

- Argoproj (or more commonly Argo) is a collection of open source tools for Kubernetes to run workflows, manage clusters, and do GitOps in Kubernetes.
- This includes **Argo Workflows**, **Argo CD**, Argo Events, and Argo Rollouts.
- https://argoproj.github.io/
- https://github.com/argoproj

What is Argo?

- **Argo Workflows**
  - Kubernetes-native workflow engine supporting DAG and step-based workflows
- **Argo CD**
  - Declarative continuous delivery with a fully-loaded UI
- **Argo Rollouts**
  - Advanced Kubernetes deployment strategies such as Canary and Blue-Green made easy
- **Argo Events**
  - Event based dependency management for Kubernetes.

Argo Workflows

- Argo Workflows is an open source container-native workflow engine for orchestrating jobs on Kubernetes.
- Argo Workflows is implemented as a Kubernetes CRD (Custom Resource Definition).
  - A ***resource*** is an endpoint in the Kubernetes API that stores a collection of API objects of a certain kind.
  - ***Custom resources*** are extensions of the Kubernetes API. It represents a customization of a particular Kubernetes installation.

- Create and run advanced workflows entirely on Kubernetes

- https://argoproj.github.io/argo-workflows/

Argo Workflows

- Define workflows where each **step** in the workflow is a **container**.
- Model multi-step workflows as a sequence of tasks or capture the **dependencies between tasks** using a directed acyclic graph (**DAG**).
- Easily run **compute intensive jobs** for machine learning or data processing using Argo Workflows on Kubernetes.
- **Run CI/CD pipelines** natively on Kubernetes without configuring complex software development products.

Main Features

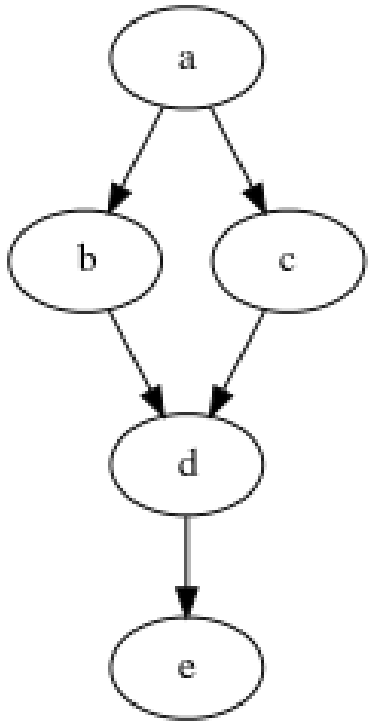
- Fully featured UI
- Templating and composability
- Workflow archive
- Cron Workflows
- REST API

Main concepts in Argo

- Workflow
  - the sequence of processes through which a piece of work passes from initiation to completion
- Workflow template
  - A workflow that is persisted on the cluster
  - Can be submitted as a whole or referenced in part by other workflows or workflows template

How Argo Works

- Argo adds a new object to Kubernetes called a Workflow, that we can create and modify as any other Kubernetes object (like a Pod or Deployment). A Workflow is, in fancy speak, a directed acyclic graph of “steps”.



How Argo Works

With Argo, **each “step” executes in a pod** and can run in parallel with, or as a dependency of, any number of other steps.

- Some of Argo's features include:
  - parametrization and conditional execution
  - passing artifacts between steps
  - timeouts and retry logic
  - recursion and flow control
  - suspend, resume, and cancellation
  - memoized resubmission

How Argo Works

- Why do we want to use Argo? Why not use another tool like Airflow, or hack something up on our existing Jenkins cluster?
- Because Kubernetes!
- Argo doesn't reinvent what Kubernetes already provides. If we know how to attach a volume to a pod, we know how to attach a volume to a step in our workflow. The same applies to networking, environment variables, resource requests/limits, service accounts, node/pod (anti-)affinities, and everything else a pod can define.
- This is possible because Workflows use the same mechanism as vanilla Kubernetes Deployments or DaemonSets. For example, they use a pod template.

A Simple Workflow

- That said, let's take a brief look at possibly the simplest workflow object

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow                    # new type of k8s spec
metadata:
  generateName: hello-world-      # name of the workflow spec
spec:
  entrypoint: whalesay            # invoke the whalesay template
  templates:
  - name: whalesay                # name of the template
    container:
      image: docker/whalesay
      command: [cowsay]
      args: ["hello world"]
```

A Simple Workflow

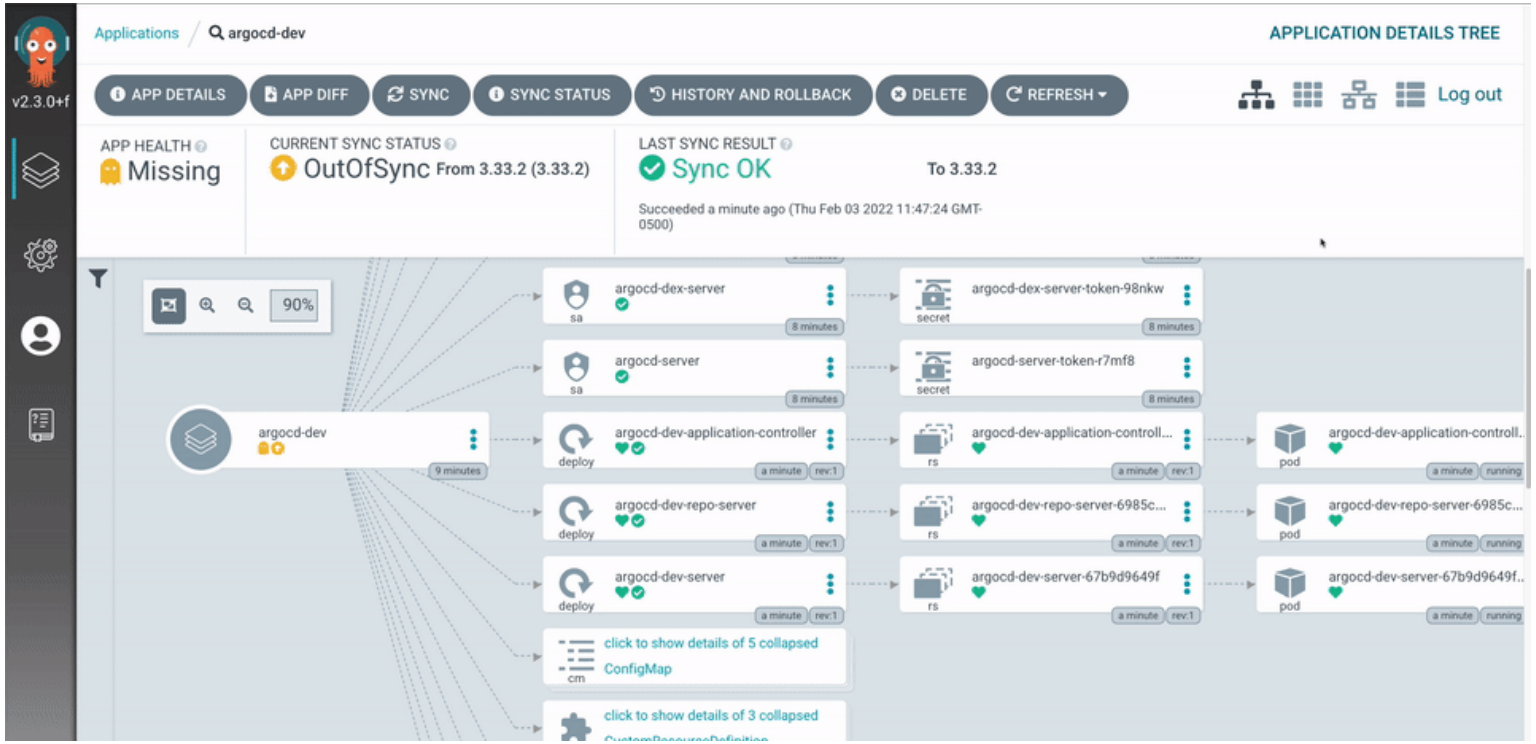
- This workflow is largely analogous to running the following command locally:

```
$ docker container run docker/whalesay cowsay "hello world"
```

- While this example isn't very exciting – fear not! Workflows can quickly get complicated to suit our needs. There are several examples of the features listed above in the official docs.

What Is Argo CD?

- Argo CD is a declarative, GitOps continuous delivery tool for Kubernetes.



- Argo CD **automates the deployment** of the desired application states in the specified target environments.
- Application deployments can **track updates** to branches, tags, or pinned to a specific version of manifests at a Git commit.
- Argo CD follows the **GitOps** pattern of using Git repositories as the source of truth for defining the desired application state.