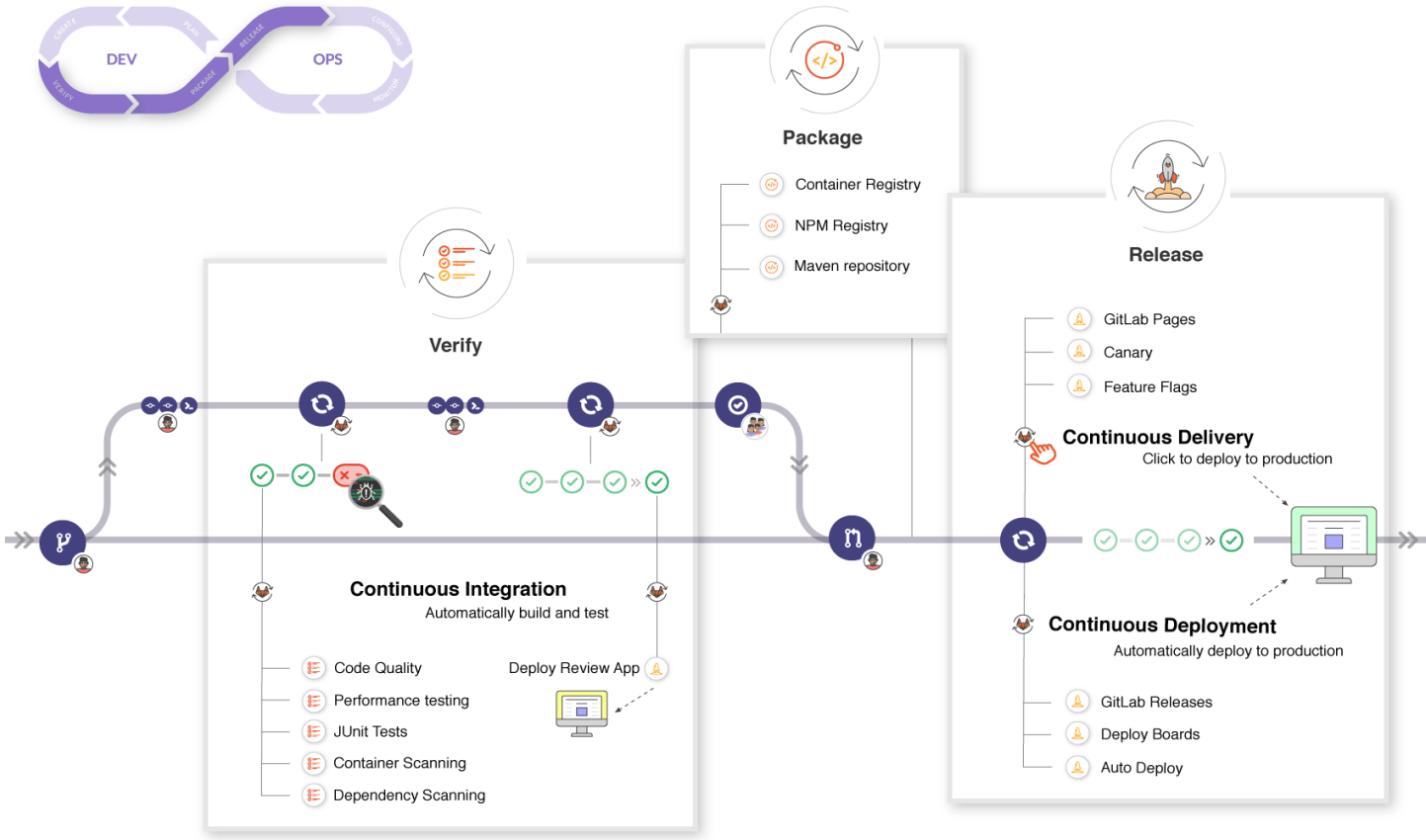
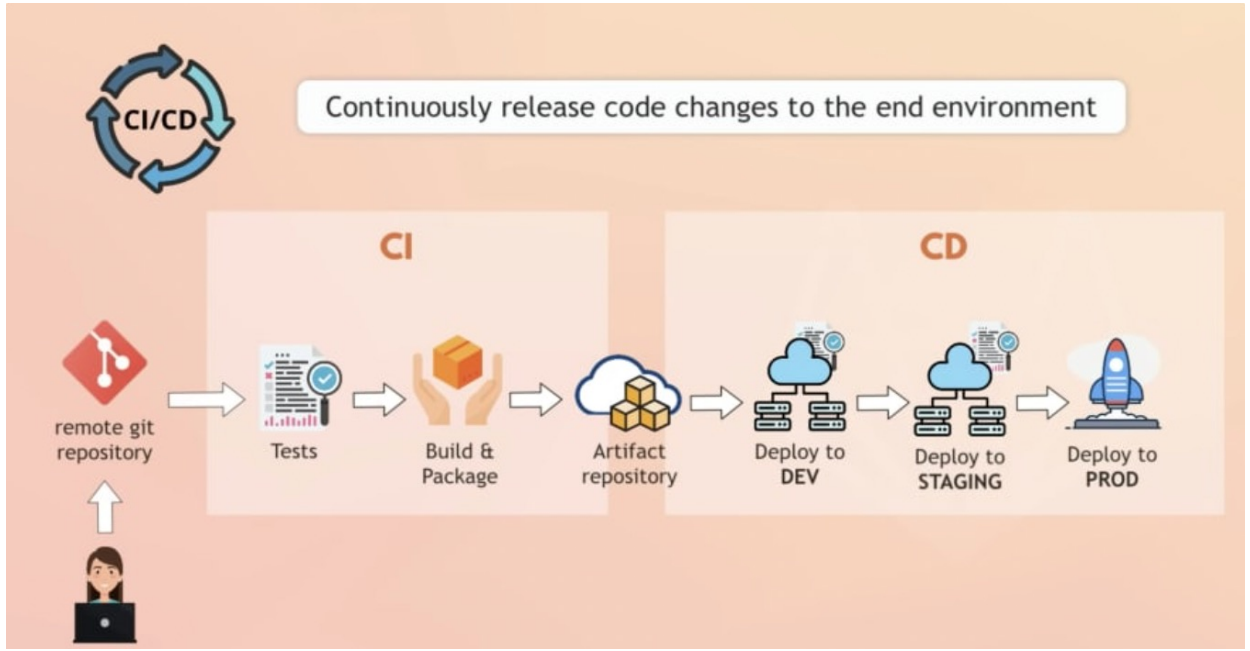
What is GitLab CI/CD?

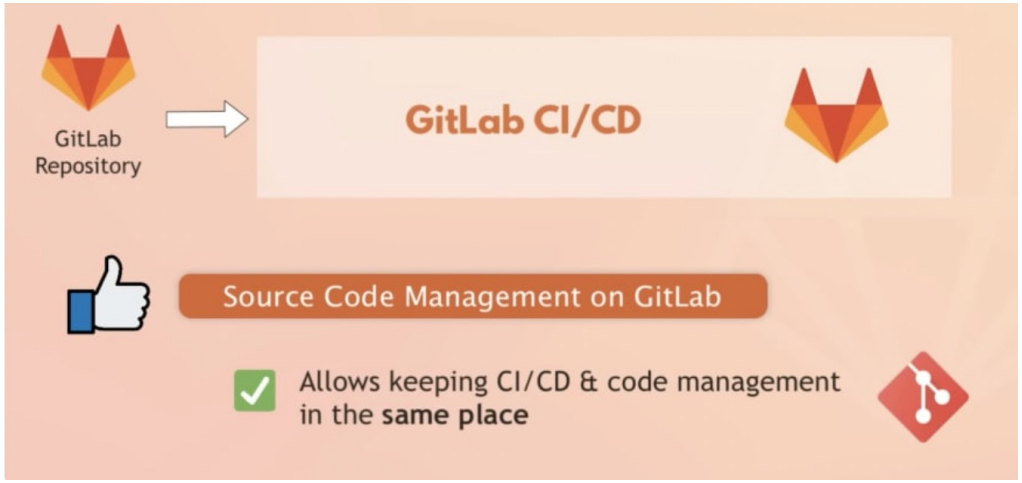
- GitLab CI (Continuous Integration) service is a part of GitLab that build and test the software whenever developer pushes code to application.
- GitLab CD (Continuous Deployment) is **a software service that places the changes of every code in the production which results in everyday deployment of production**.



Many CI/CD platforms

Source Code & CI/CD on same platform

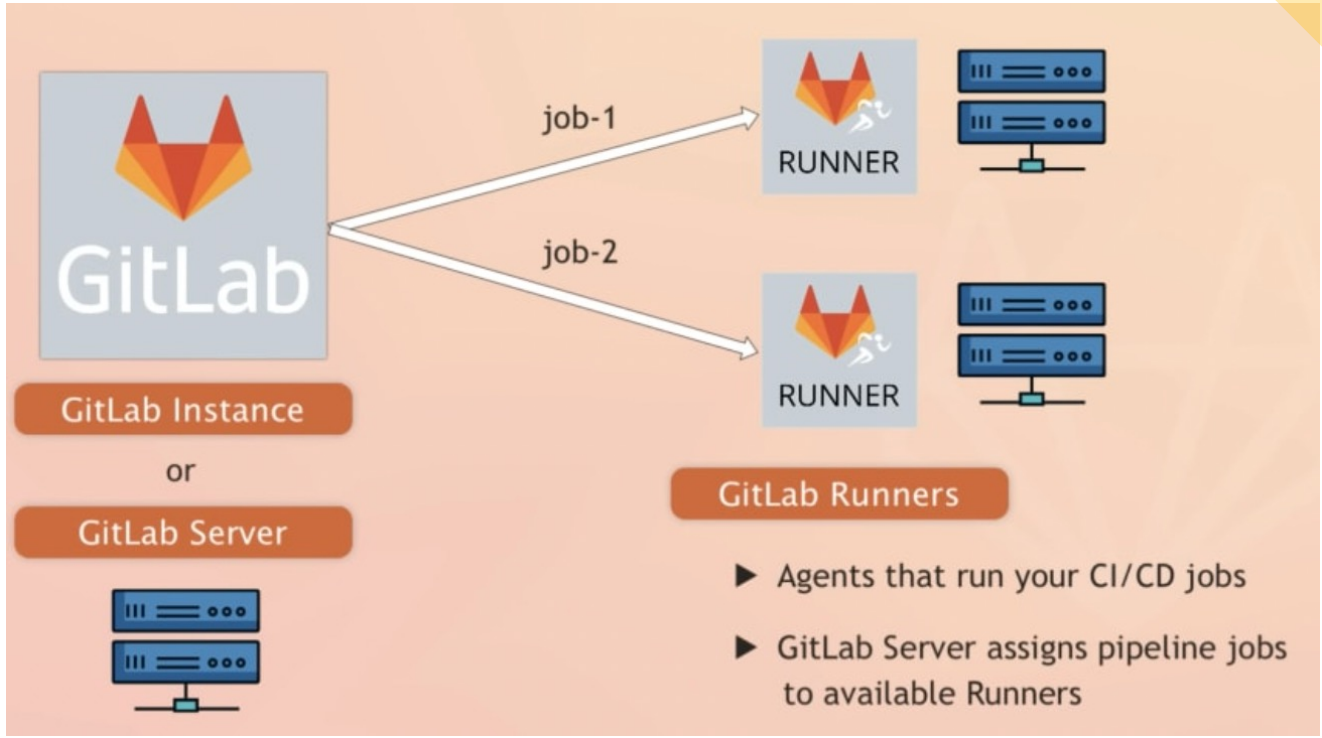




GitLab Architecture

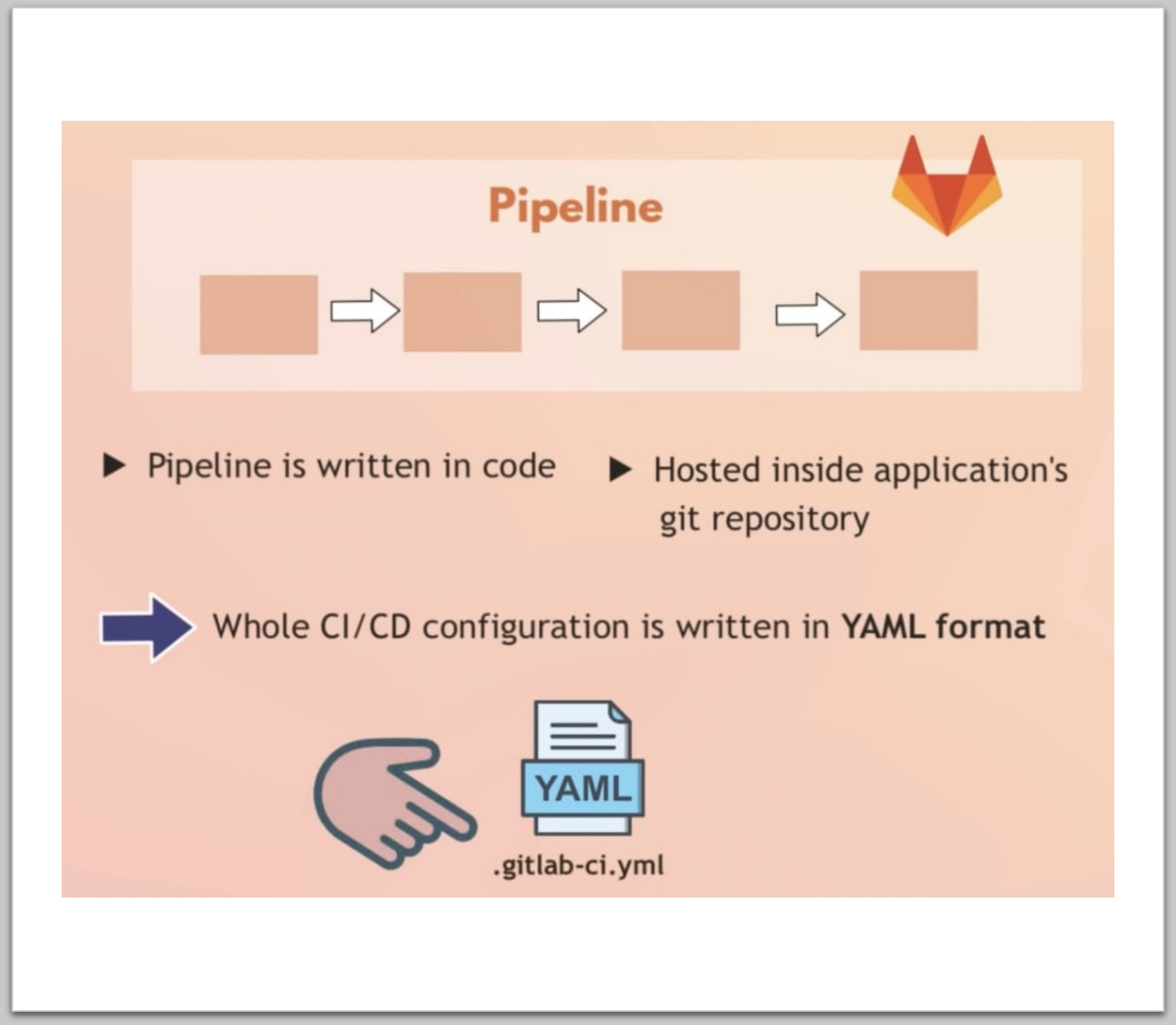
- Gitlab instance or also called Gitlab server that **hosts your application code and your pipelines and basically the whole configuration**.

- **GitLab Runners**
- Gitlab instance will have multiple Gitlab runners, which are separate machines connected to the Gitlab server machine, which are actually the ones executing the pipelines.



Pipeline Configuration - .gitlab-ci.yml

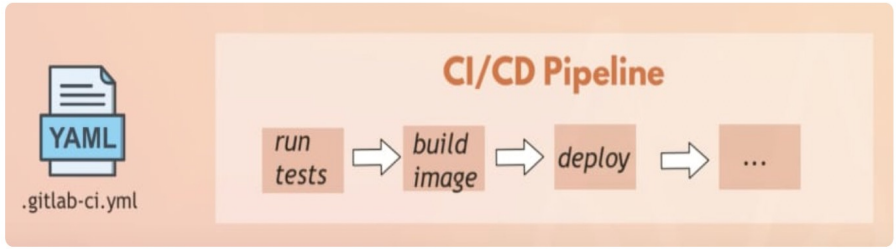
- **Configuration as Code** or **Pipeline as Code**, the whole pipeline will be written in code and hosted in the applications git repository itself in a **simple YAML file**.
- The file has to be called **.gitlab-ci.yml**, so that Gitlab can automatically detect that pipeline code and execute it without any extra configuration effort.

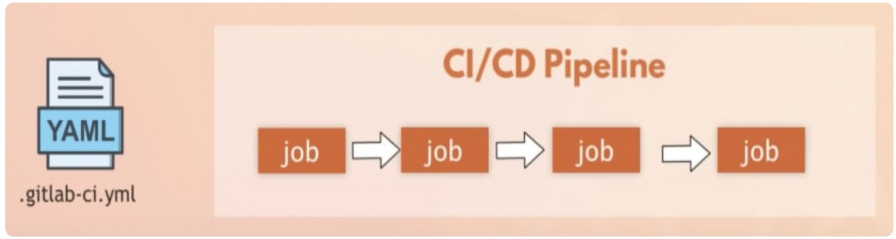


In the **root of the project's repository**,
will create this YAML file and we will
write all the pipeline configuration inside and we can actually do that directly in the Gitlab UI as well, so we don't have to switch back and forth from the editor to Gitlab

Jobs

- The tasks in the CI/CD pipeline such as running tests, building an image, deploying to a server etc. are configured as called **'jobs'**.
- Each job has a name and inside the job we have a couple of parameters or a couple of attributes or things that we want to configure for the job.
- The first attribute and the only required attribute of a job is what's called a **script**. And script is basically, where we list any commands that should be executed for that job:



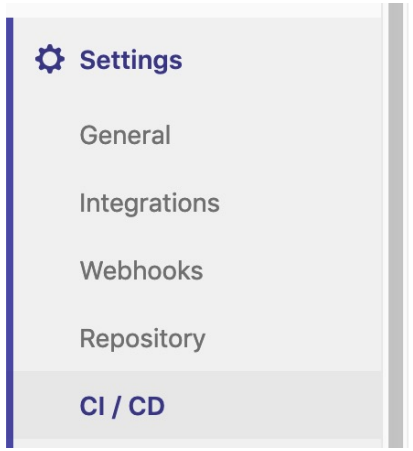


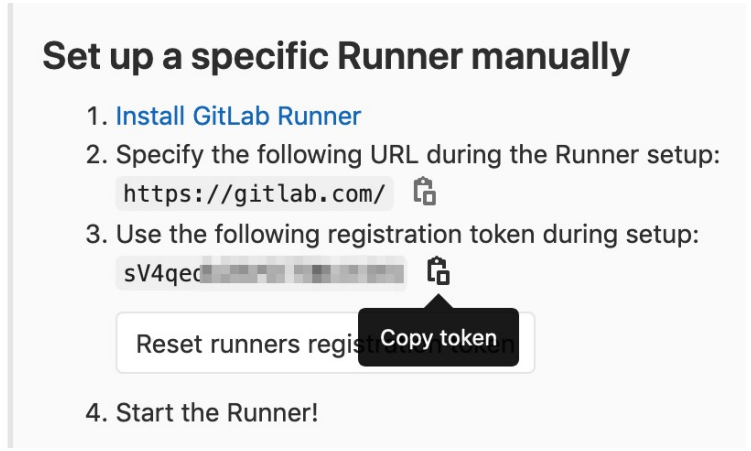

```
job1:
  script: "execute-script-for-job1"

job2:
  script: "execute-script-for-job2"
```

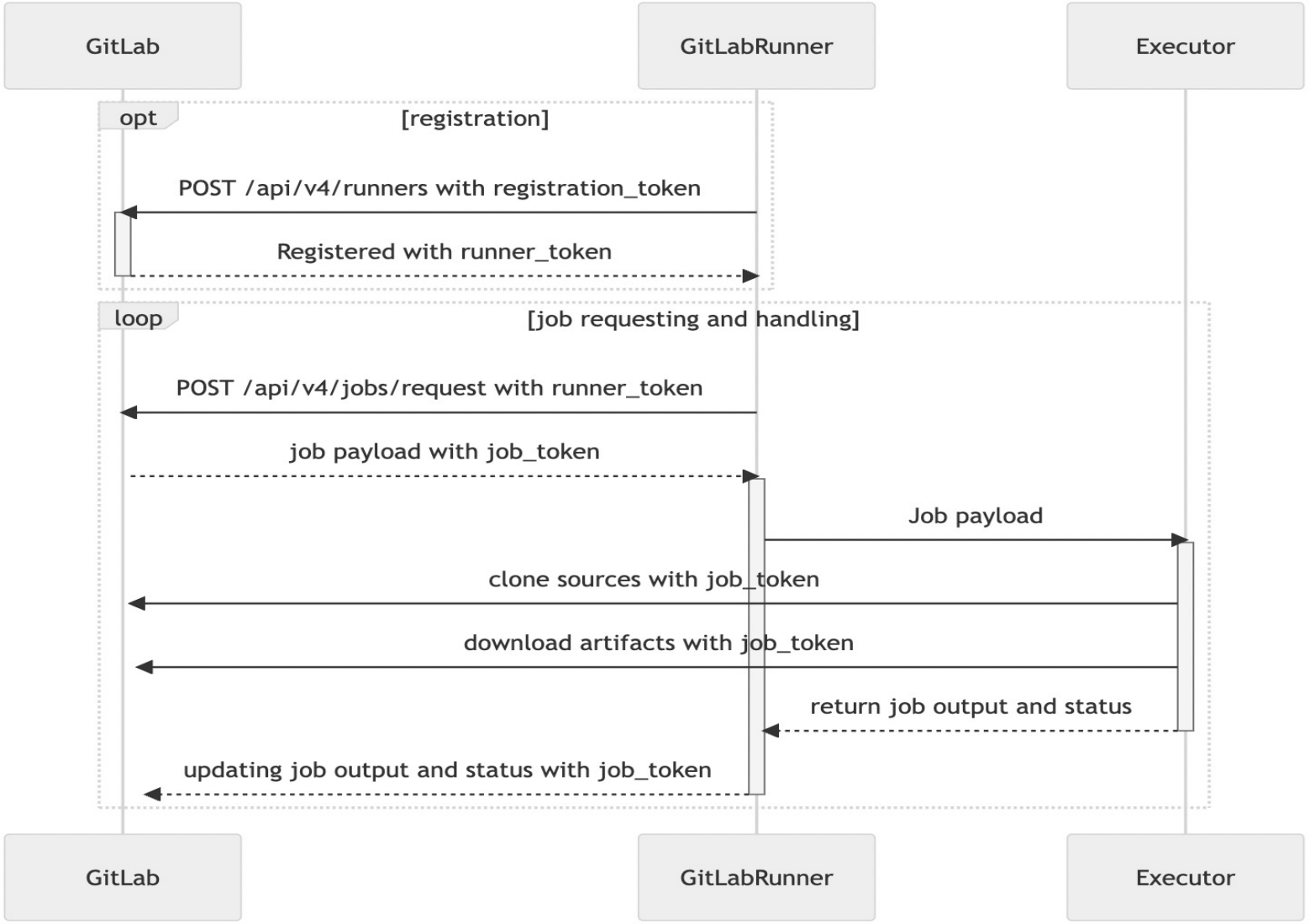
Runner Setup - Shell

- curl -L https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.deb.sh > script.deb.sh
- sudo bash script.deb.sh
- sudo apt install gitlab-runner
- systemctl status gitlab-runner
- For REGISTRATION_TOKEN go to Project settings -> CICD -> Runners -> Copy the token and paste in below command
- sudo gitlab-runner register --url https://gitlab.com/ --registration-token $REGISTRATION_TOKEN





Runner execution flow



Executors

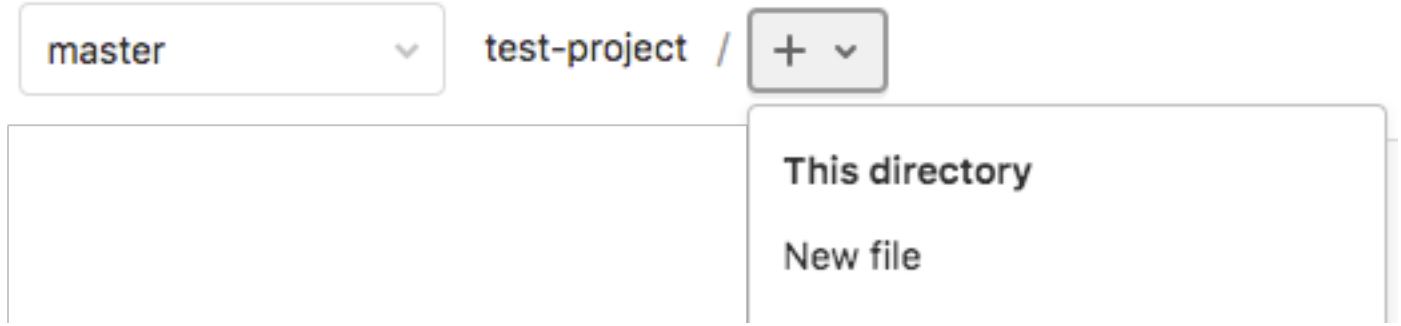
- GitLab Runner implements a number of executors that can be used to run your builds in different scenarios.

Types of Runners:

1) Shell
2) Docker
3) Kubernetes

To create a .gitlab-ci.yml file:

1. On the left sidebar, select **Repository > Files**.
2. Above the file list, select the branch you want to commit to. If you're not sure, leave master or main. Then select the plus icon () and **New file**:
3. For the **Filename**, type .gitlab-ci.yml and in the larger window, paste this sample code:



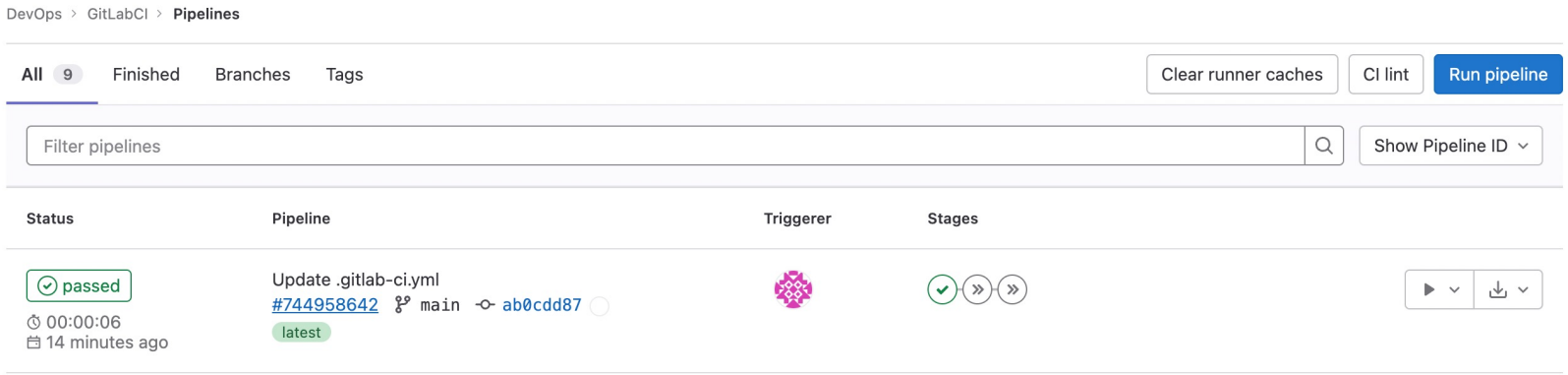

```
build-job:
  stage: build
  script:
    - echo "Hello, $GITLAB_USER_LOGIN!"

test-job1:
  stage: test
  script:
    - echo "This job tests something"

test-job2:
  stage: test
  script:
    - echo "This job tests something, but takes more time than test-job1."
    - echo "After the echo commands complete, it runs the sleep command for 20 seconds"
    - echo "which simulates a test that runs 20 seconds longer than test-job1"
    - sleep 20

deploy-prod:
  stage: deploy
  script:
    - echo "This job deploys something from the $CI_COMMIT_BRANCH branch."
  environment: production
```

View the status of your pipeline and jobs




```
1  Running with gitlab-runner 15.7.1 (6d480948)
2    on test DYmn9yg-
3  Resolving secrets
5  Preparing the "shell" executor
6  Using Shell executor...
8  Preparing environment
9  Running on ip-172-31-22-197...
11 Getting source from Git repository
12 Fetching changes with git depth set to 20...
13 Initialized empty Git repository in /home/gitlab-runner/builds/DYmn9yg-/0/devops3415/GitLabCI/.git/
14 Created fresh repository.
15 Checking out da4ffdcc as main...
16 Skipping Git submodules setup
18 Executing "step_script" stage of the job script
19 $ echo "Hello this is a test pipeline"
20 Hello this is a test pipeline
22 Cleaning up project directory and file based variables
24 Job succeeded
```