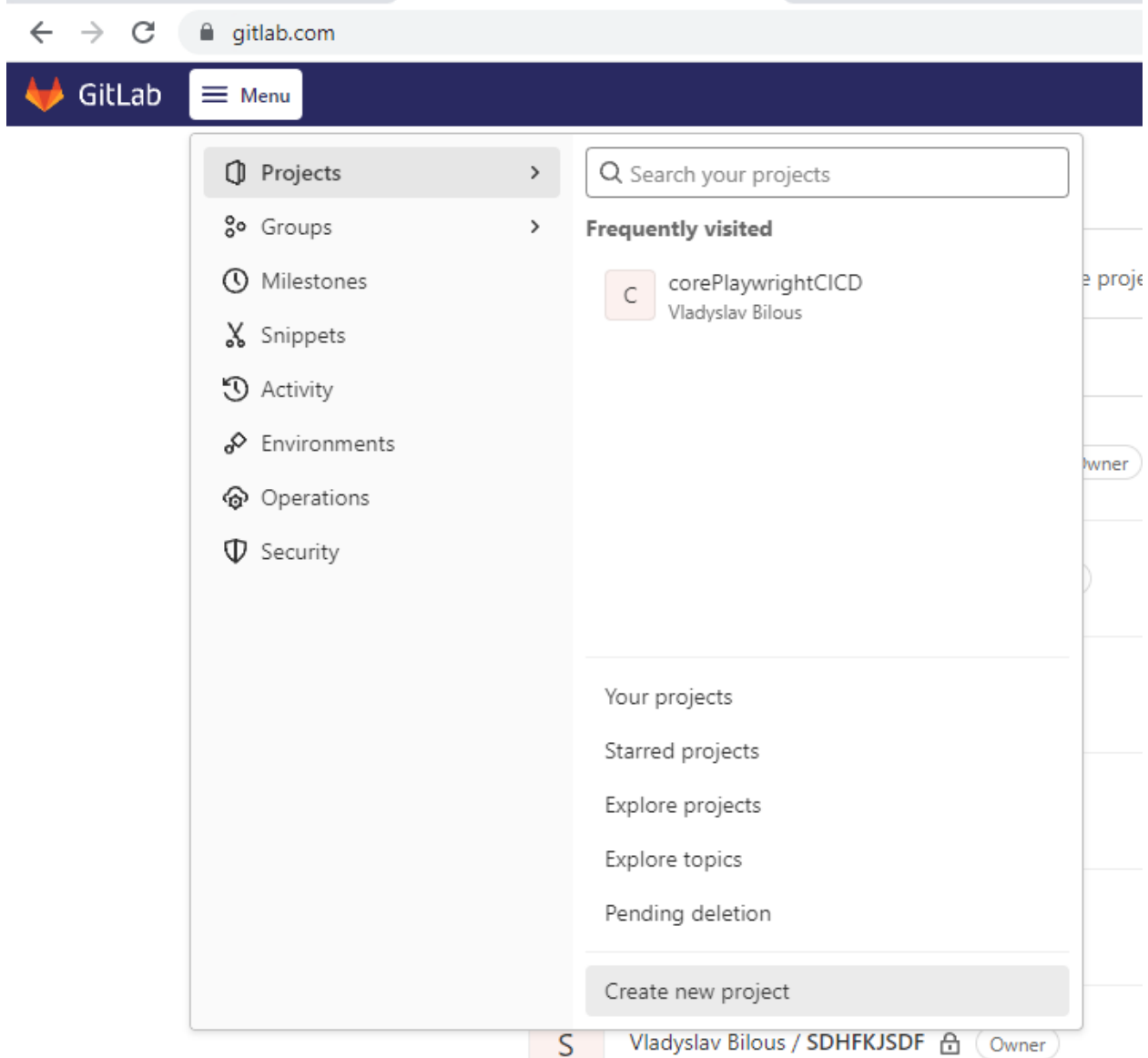


Let's create a simple project for GitLab to learn how to use Pipelines, create .yml file and deploy reports to Pages.

For this guide you will need to create a GitLab account. <https://gitlab.com/>

After logging in, you will be able to create new project by clicking Menu button – Create new project



There are a lot of different ways to proceed, let's import a repository from GitHub to see how does it work. First of all we will need to create a project and publish it to GitHub.

Let's use simple test based on Playwright framework.

Create a folder and install Playwright:

```
npm init playwright@latest
```

Also install Allure:

```
npm i -D allure-playwright
```

Now we have installed everything that we needed, test was autogenerated and we are ready to publish it to GitHub. But before let's do some additional changes into our .gitignore file adding allure-results and reports. Now our gitignore file looks like:

```
❖ .gitignore
1  node_modules/
2  test-results/
3  playwright-report/
4  allure-results
5  allure-report
6
```

Now let's run autogenerated tests using next command:

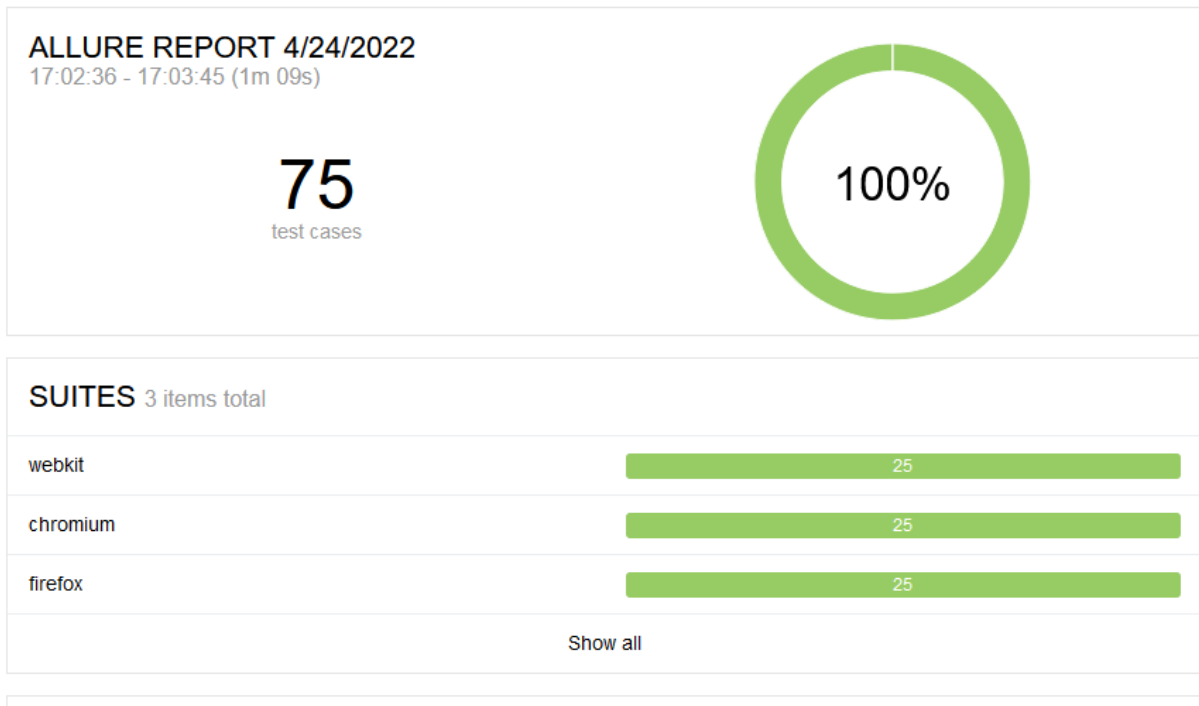
```
npx playwright test --reporter=line,allure-playwright
```

and check that everything works fine:

```
allure generate
```

```
allure serve
```

after it you should be able to see

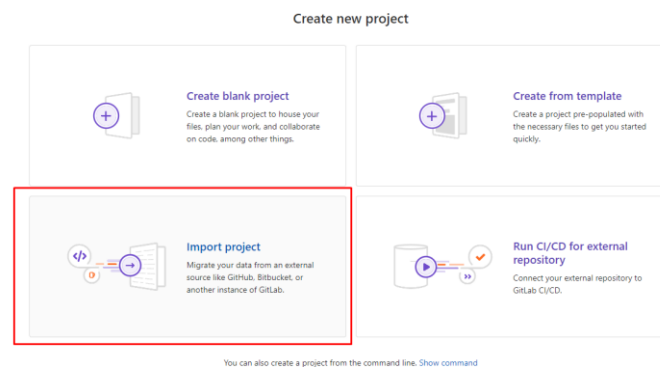


lets create some scripts in package.json to make our process easier:

```
"scripts": {  
  "test": "npx playwright test --reporter=line,allure-playwright",  
  "allure-generate": "allure generate --clean",  
  "allure-serve": "allure serve"  
},
```

Now just upload the project to GitHub

Let's return to GitLab and do an import of project:

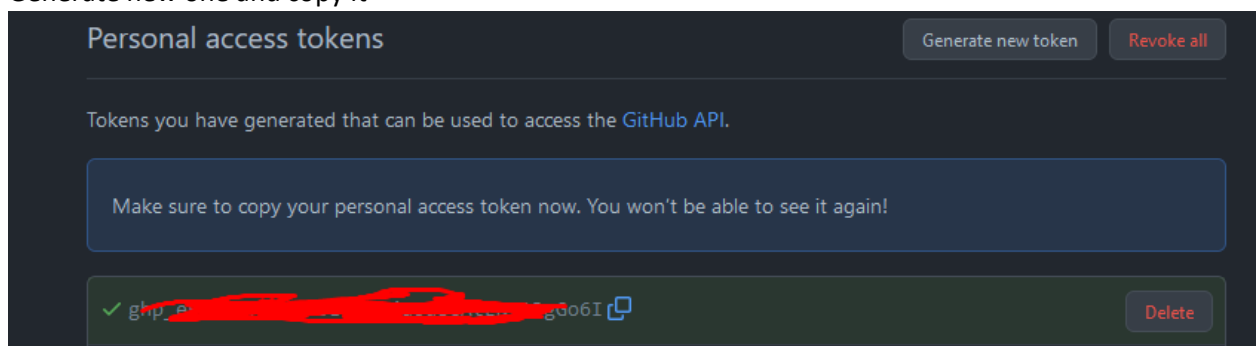


Select import project

And then "GitHub".

For that action you will need to provide an access to your GitHub data, let's do it using Personal Token:

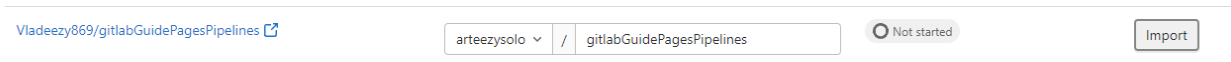
1. Go to GitHub -> Settings -> Developer Settings -> Personal access tokens
2. Generate new one and copy it



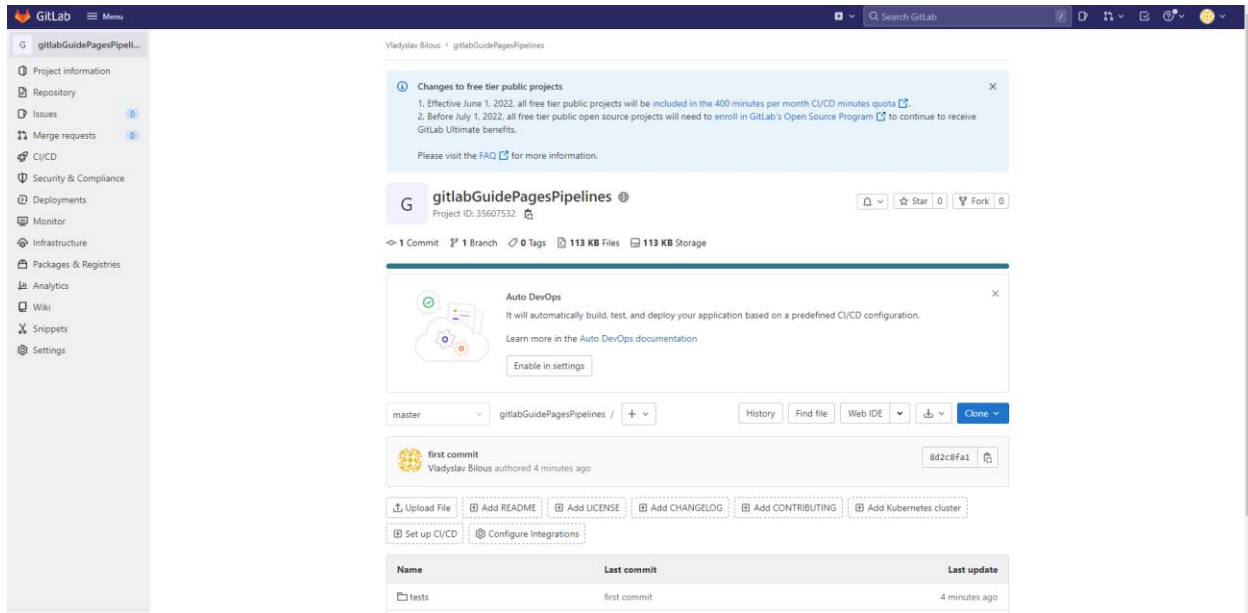
Then just use it on GitLab importing page

Now you are able to import your any project from GitHub to GitLab

Find needed project and import it

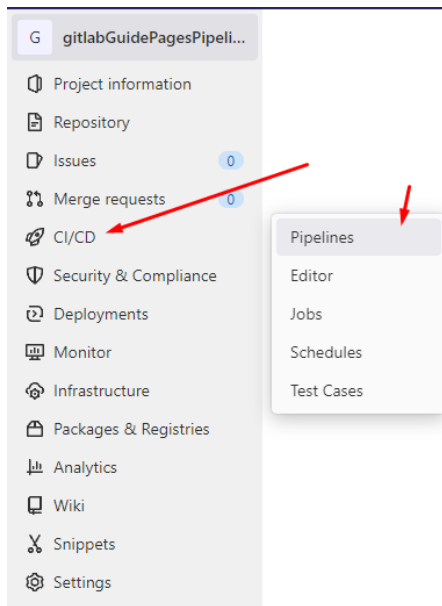


You will see

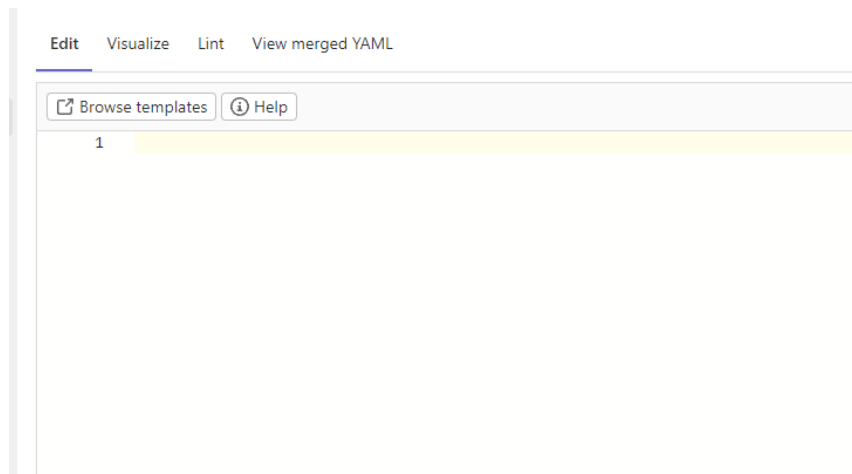


Okay, now we need to create .yaml file.

Go to CI/CD -> Pipelines



Try Test Template and delete everything from here:



Lets build yml file step by step to understand everything whats going on here.

First of all we need to divide all logical processes into parts like: testing and deploy, to do it, just create 2 stages (do not copy it now, the full code will be bellow, for now try to understand how does it work)

```
stages:
  - testing
  - deploy
```

Then we need to create a thing called “Job” which is responsible for steps our Pipeline will go, first one will be called docker_job because firstly we need to install our docker image

```
docker_job:
  stage: testing
  tags:
    - docker
  image: atools/chrome-headless:java11-node14-latest
```

Then create a before_script section, which will be used for pre-installing necessary components

```
before_script:
  - npm ci
  - npx playwright install
  - npm install allure-commandline --save-dev
```

Here we will install node, playwright and allure.

Next will be script section, here we will run our tests

As you remember we have created scripts in package.json, now we will use them

```
script:
  - npm run test
```

Now we will create after_script part, which will be generating our allure results

```
after_script:
  - npx allure generate allure-results
```

And add:

```
allow_failure: true
artifacts:
  when: always
  paths:
    - ./allure-report
  expire_in: 1 day
```

allow_failure will be responsible for letting job fail if some tests will fail and continue working

artifacts will save our report parts to use them in our Allure Pages.

And last one part of yml file is deploying our artifacts to gitlab pages to give us an opportunity to see Allure

```
pages:
  stage: deploy
  script:
    - mkdir public
    - mv ./allure-report/* public
  artifacts:
    paths:
      - public
  rules:
    - when: always
```

By default the folder for Pages files is “public”, on the docker stage it will be deleted, so we need to create it and move all results to this folder.

Now we have next yml file:

```
stages:
  - testing
  - deploy

docker_job:
  stage: testing
  tags:
    - docker
  image: atools/chrome-headless:java11-node14-latest
  before_script:
    - npm ci
```

```

- npx playwright install
- npm install allure-commandline --save-dev
script:
- npm run test
after_script:
- npx allure generate allure-results
allow_failure: true
artifacts:
  when: always
  paths:
    - ./allure-report
  expire_in: 1 day
pages:
  stage: deploy
  script:
    - mkdir public
    - mv ./allure-report/* public
  artifacts:
    paths:
      - public
  rules:
    - when: always

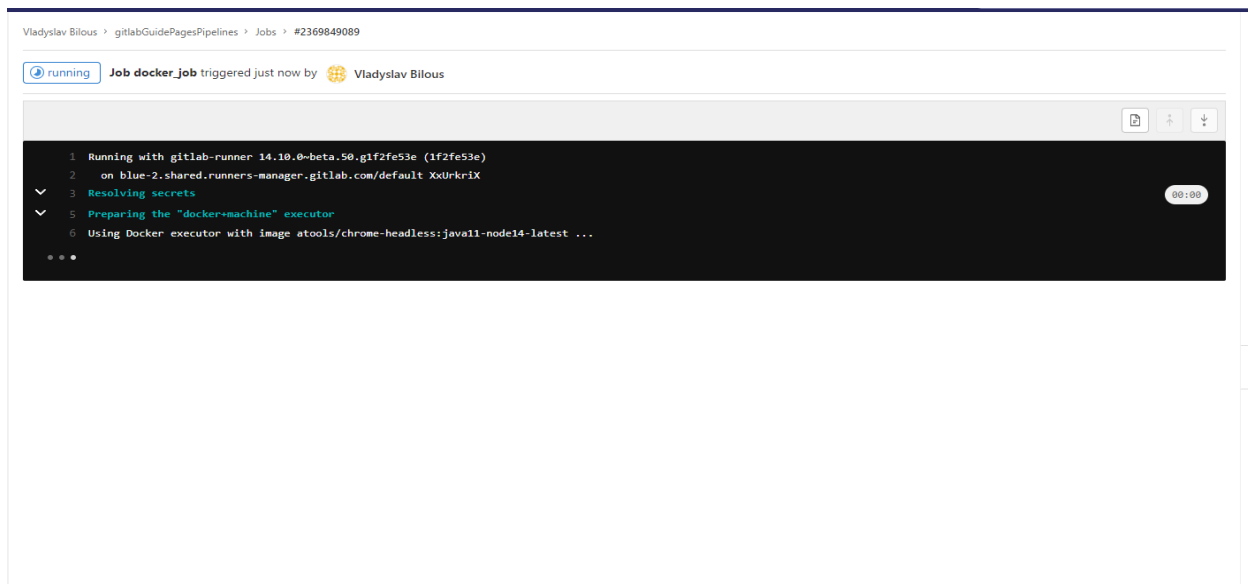
```

Pay attention to every single space, the whole yml file can be not working if you will do something wrong.

Now save it and you will be able to see



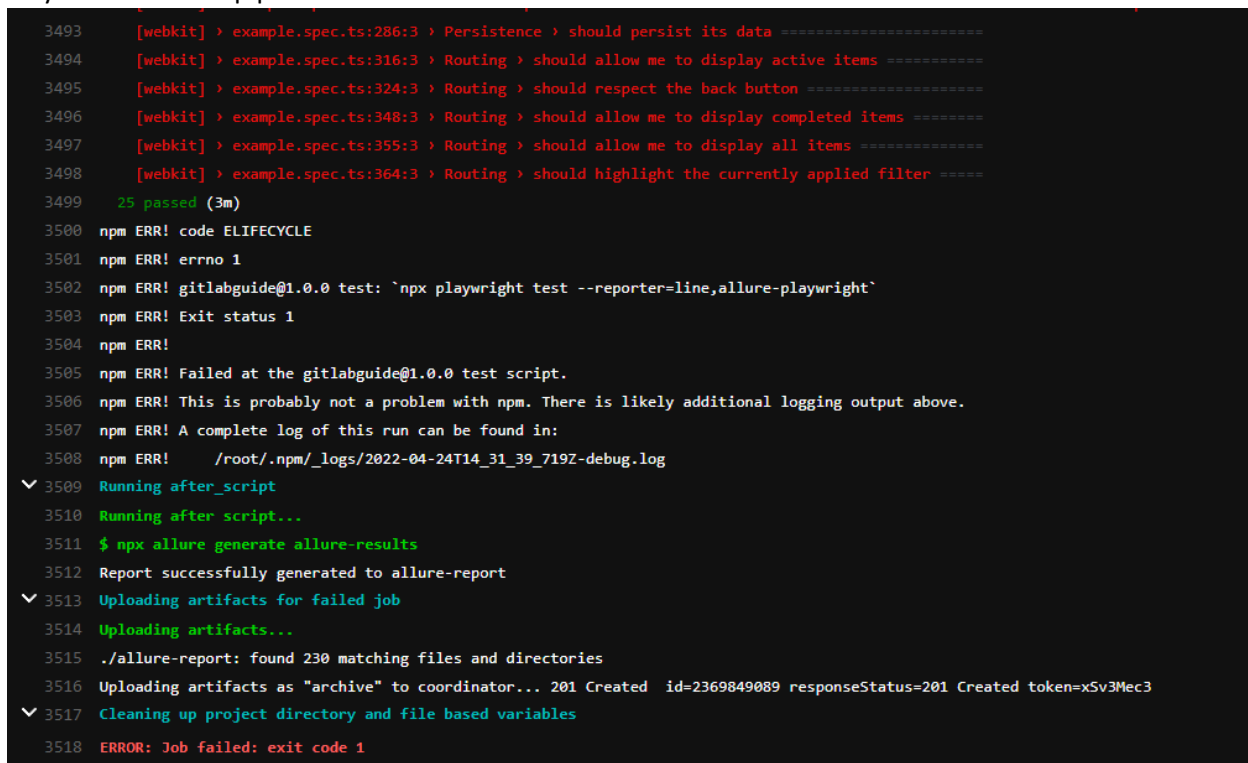
Click view pipeline to check how does it go.



All tests that were run on Chromium will be passed, the rest – will be failed because we will need to install additional libraries for working with Firefox etc.

But we need to see how does it work so let's skip these errors:

As you can see the pipeline is finished



Now go to Pages and check the result:

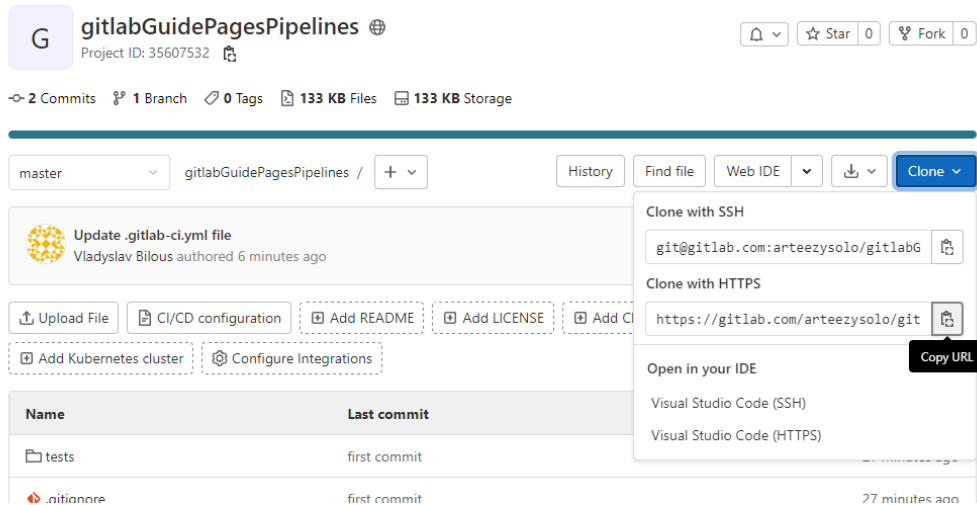
The screenshot shows the GitLab settings interface. On the left, a sidebar menu lists various settings categories: Project information, Repository, Issues (0), Merge requests (0), CI/CD, Security & Compliance, Deployments, Monitor, Infrastructure, Packages & Registries, Analytics, Wiki, Snippets, Settings, General, Integrations, Webhooks, Repository, CI/CD, Monitor, Pages, Packages & Registries, and Usage Quotas. Red arrows point to the 'Settings' and 'Pages' items in this menu. The main content area is titled 'Pages' and includes a search bar, a description of GitLab Pages, a checkbox for 'Force HTTPS (requires valid certificates)' which is checked, and a 'Save changes' button. Below this, a section titled 'Access pages' shows 'Your pages are served under:' followed by the URL 'https://arteezysolo.gitlab.io/gitlabGuidePagesPipelines' with an external link icon. A red arrow points to this URL. At the bottom, a red bar contains the text 'Remove pages', and below that, a message states 'Removing pages will prevent them from being exposed to the outside world.' with a 'Remove pages' button.

As you can see, all reports are located here, so, everything is working fine.

Actually that's the end, bellow I will show some features for continuing working with that project.

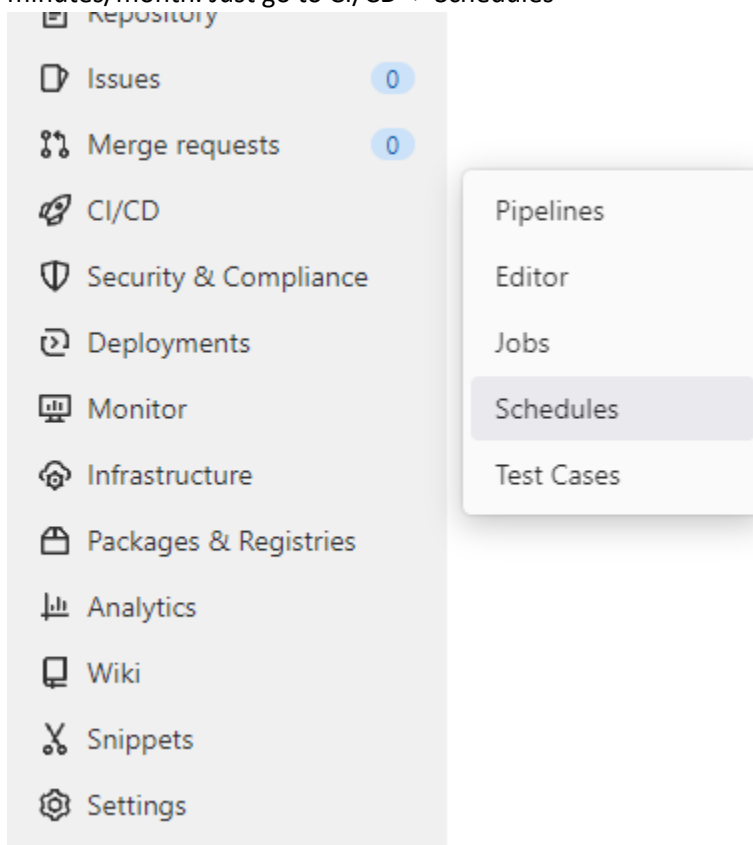
How to work now only with that GitLab project, because I have imported it from GitHub?

Not the best way but we can simply do it by deleting our project locally from our PC and do a clone from GitLab, after it, it will be auto linked to our Gitlab project and all pushed etc will be redirected to current repository.



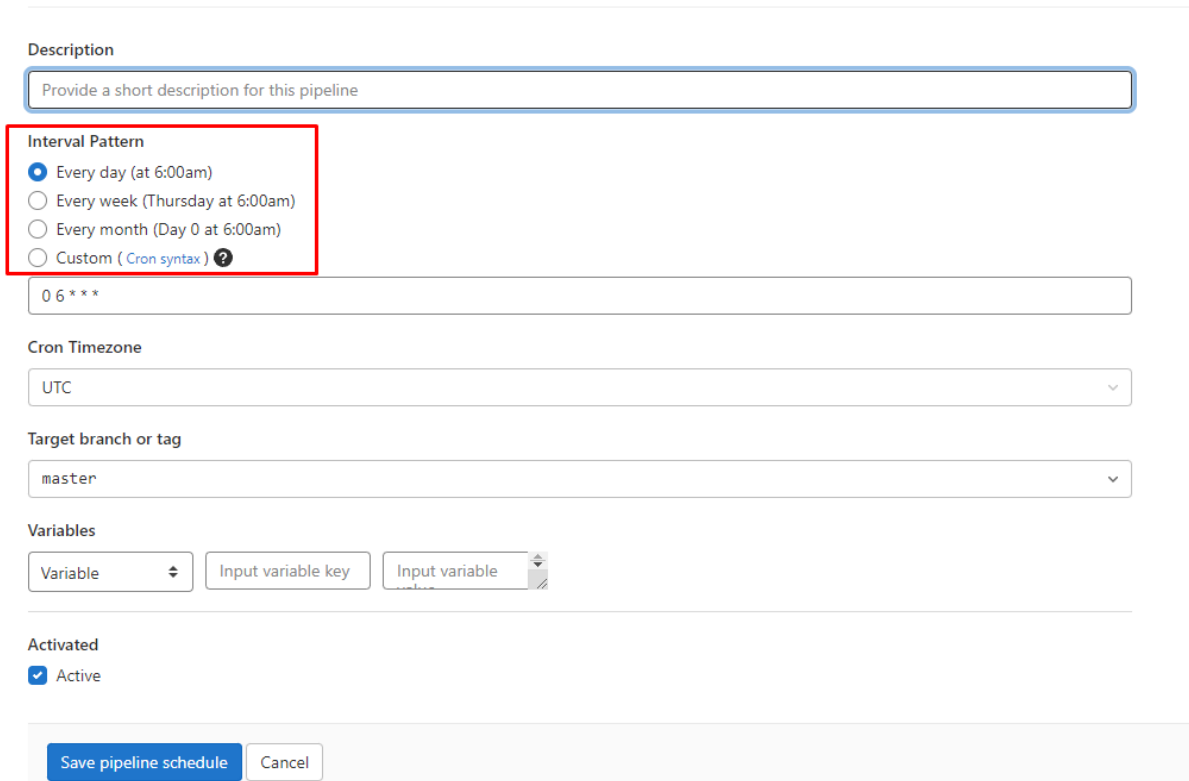
I want to run my tests every day, how to do it?

In GitLab you are able to create any schedule you want, but be careful, you have only 400 testing minutes/month. Just go to CI/CD -> Schedules



And create your own or use a template.

If you need something specific like running it every hour or 1.5 hours, you will need to use Cron syntax for that goal



Description

Provide a short description for this pipeline

Interval Pattern

- ☒ Every day (at 6:00am)
- ☐ Every week (Thursday at 6:00am)
- ☐ Every month (Day 0 at 6:00am)
- ☐ Custom (Cron syntax) ?

0 6 * * *

Cron Timezone

UTC

Target branch or tag

master

Variables

Variable Input variable key Input variable value

Activated

☒ Active

Save pipeline schedule Cancel

1 hour in Cron syntax for example is `0 * * * *`, google how does it work if you need something other.

When I just started using GitLab I have faced with a problem. Client provided me with account which allows me to get access to their own workspace. The problem was that this repository didn't have "Pages" page and Pipelines were not working at all because of not configured Runner for it. If you will have the same problem in future, tell client's Devops to solve this problem, it's his job.

Good Luck!