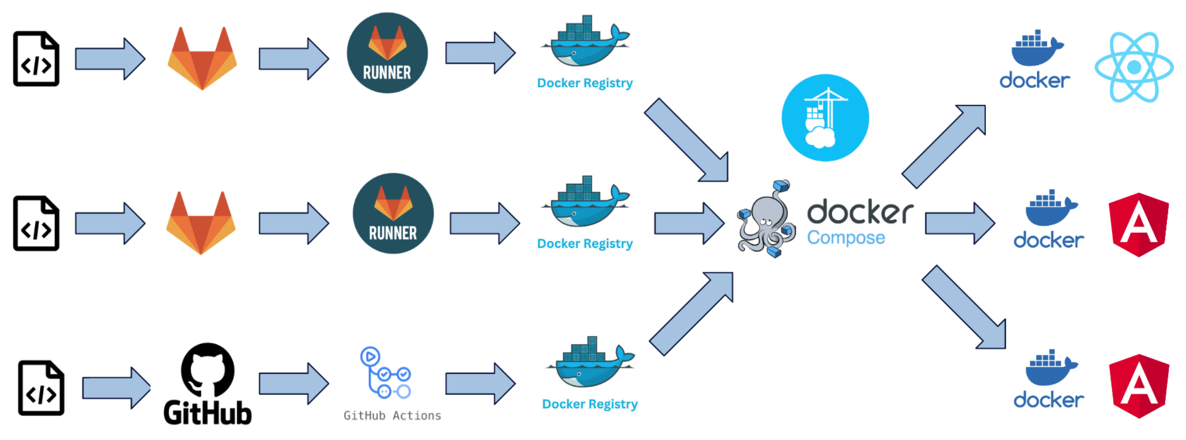
TP05 - Docker Compose and Portainer

Please read the whole subject …

An LLM model has been deployed on `gprrrr.co/ollama` (`http://62.72.19.6/ollama`)

The objective is to create a small frontend to consume the LLM API, version it with Github or Gitlab, build a Docker image in Gitlab container registry, then deploy it in a docker-compose stack.



A. Small Dockerized frontend

Should feature whatever you want, but it has to consume the ollama API on `gprrrr.co/ollama`.

Send a request to Ollama :

```
POST http://gprrrr.co/ollama/api/generate
     {
          "model": "mistral", // installed model, mandatory
          "prompt": "Why is the sky blue?", // your prompt
          "stream": false // wait for the response to be complete
          before answering. You can use false, and implement
          streamed responses.
     }
```

Your application must have a path, it cannot be “localhost:4200” only, it has to have a path, like `localhost:4200/<yourname>` for example.

Attention : If two groups have the same path, you will have conflicts between you. (no worries if you have the same port)

B. If you choose Gitlab : Gitlab Container Registry

You have to create a job to build your Docker image.
Example of a script for building and pushing into container registry :

```
image: docker:latest

build-docker-image:
 services:
   - docker:dind
 before_script:
   - DATE=`date +"%d%m%Y"`
   - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD
$CI_REGISTRY
 script:
   - docker build -t
$CI_REGISTRY/<gitlab-group>/<repo>/<image-name>:$DATE -t
$CI_REGISTRY/<gitlab-group>/<repo>/<image-name>:latest .
   - docker push $CI_REGISTRY/<gitlab-group>/<repo>/<image-name>
--all-tags
```

Hint : `$CI_REGISTRY_USER, $CI_REGISTRY_PASSWORD & $CI_REGISTRY` are default gitlab variables that are already accessible inside the runner.

The Gitlab container registry is available on Group page => Deploy => Container registry

C. If you choose Github : Github Container Registry

```
...
jobs:
 build-and-push-docker-image:
   runs-on: ubuntu-latest
   steps:
     - name: Checkout code
       uses: actions/checkout@v2
     - name: Setup Docker Buildx
       uses: docker/setup-buildx-action@v3

     - name: Login to GitHub Container Registry
       run: echo "${{ secrets.GITHUB_TOKEN }}" | docker login
ghcr.io -u ${{ github.actor }} --password-stdin

     - name: Build and push Docker image
       run: |
         DATE=$(date +"%d%m%Y")
         docker buildx build --push --tag ghcr.io/${{
github.repository_owner }}/front:$DATE --tag ghcr.io/${{
github.repository_owner }}/front:latest .
```

Before pushing, go to Settings, Actions, General, Workflow permissions, check “Read and write permissions” and save.

The built images are available in your Github profile, in “Packages” tab.

D. Docker compose

Add your application with the right configuration to https://gitlab.com/isen5910803/tp05-docker-compose-architecture, test locally, **ask me for correction**, then merge into main branch.

Hint : you should see Traefik proxy page on localhost:8080, and your app on localhost/<yourname>

E. Portainer

Login to https://gprrrr.co:9443/ , with `student`, `IsenDevops2024!!!` credentials.

go to Registries : add your registry (Gitlab or Github)

If needed, create access tokens in your workspaces to be able to pull images from Portainer.

If ready, go to Stacks, TP05, and click on “pull and redeploy”

Your application should be available on http://gprrrr.co/yourpage

Well done ! Please help the other students ;)

F. Automitize the delivery

(Only one group can do that..) : You can add a ci job to the docker compose architecture repository, to call Portainer Webhook to redeploy docker compose architecture when a new docker image is built.