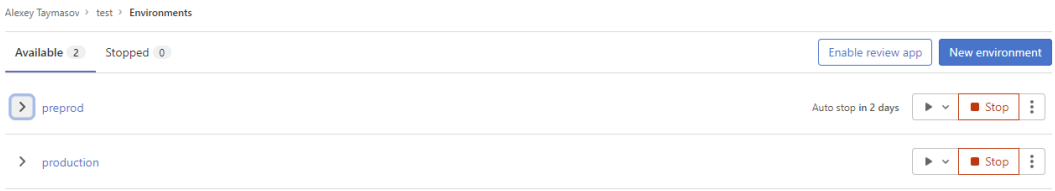
ЗАДАНИЕ № 3

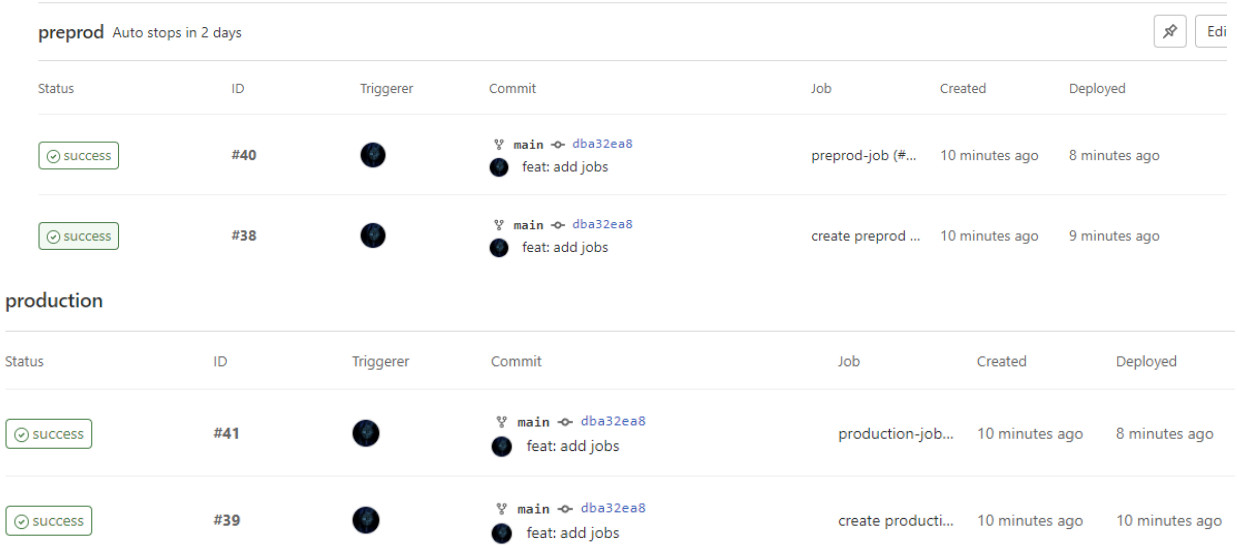
1) Добавить 2 окружения "preprod" и "production"
2) Добавить отдельные deploy job для каждой среды
3) Добавить переменную "$MyLogin" внутри .gitlab-ci.yml, которая будет меняться в зависимости от среды.
4) Добавить переменную "$MyPassword" не используя .gitlab-ci.yml, которая так же будет меняться в зависимости от среды.
* Добавить скрипт в .gitlab-ci.yml, который найдёт все запущенные pipeline по названии ветки(ref) и остановит их.

РЕШЕНИЕ:

1)
Создано 2 окружения, также окружение preprod удаляется автоматически через 2 дня или вручную.



2)
Для примера “preprod-job” взял из предыдущего задания сборку docker контейнера, а “production-job” запуск этого контейнера.



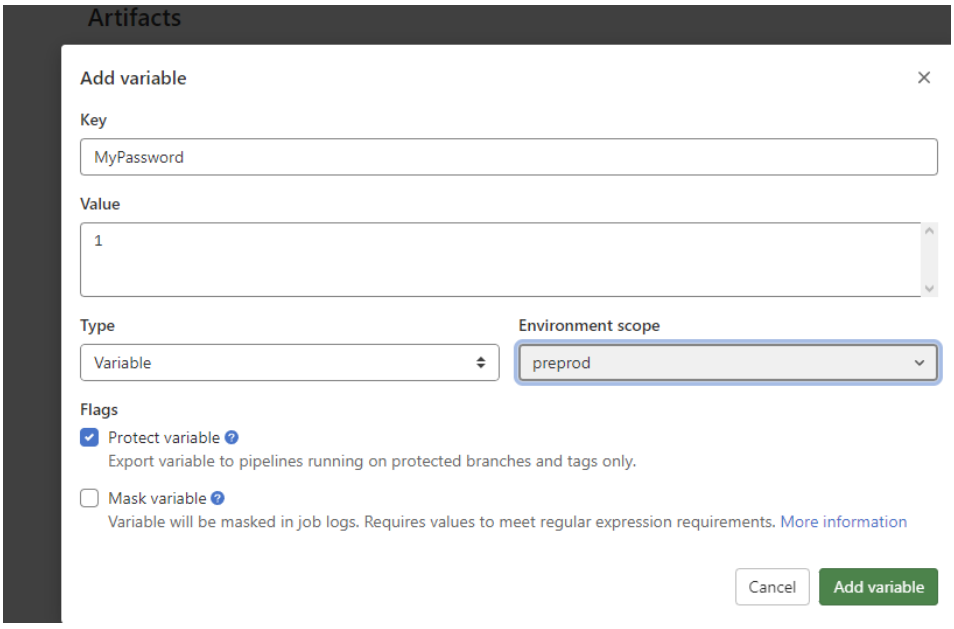

```
stages:
  - create environment
  - build
  - test
  - stop

create preprod:
 stage: create environment
 environment:
    name: preprod
    on_stop: stop preprod
    auto_stop_in: 3 day
 script:
  - echo "${CI_COMMIT_REF_NAME} was deployed on preprod"
 only:
  - main

create production:
 stage: create environment
 environment:
```

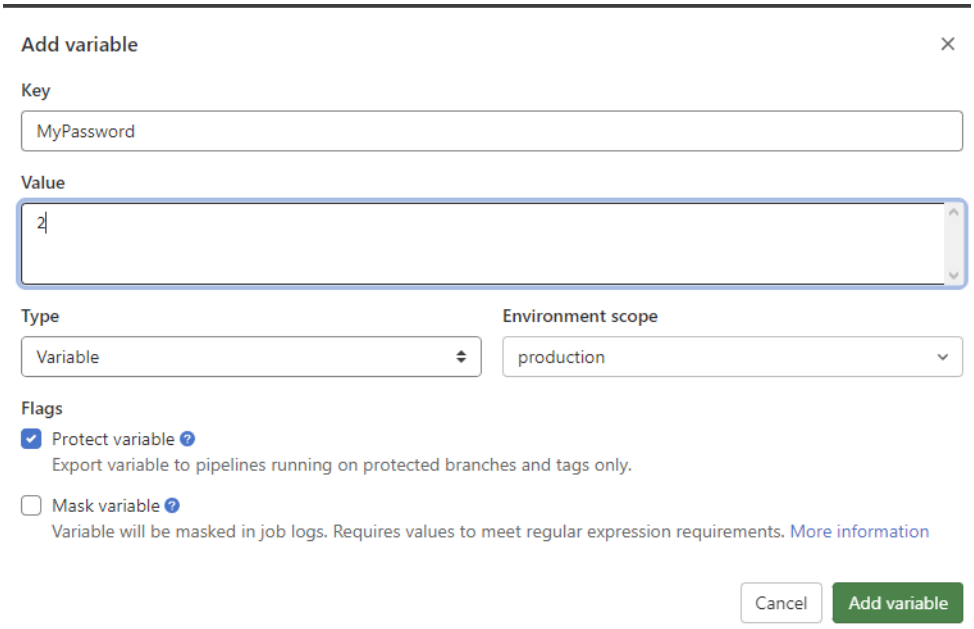
```
    name: production
  script:
   - echo "${CI_COMMIT_REF_NAME} was deployed on production"
  only:
   - main

preprod-job:
  stage: build
  environment:
    name: preprod
  image:  docker
  services:
    - docker:dind
  script:
    - echo $CI_REGISTRY_PASSWORD | docker login  -u $CI_REGISTRY_USER $CI_REGISTRY --password-stdin
    - docker build -t $CI_REGISTRY_IMAGE .
    - docker push $CI_REGISTRY_IMAGE
  only:
   - main

production-job:
  stage: test
  environment:
    name: production
  image:  $CI_REGISTRY_IMAGE
  script:
    - python --version
    - pip --version
    - pytest --version
    - echo "done"
  only:
   - main

stop preprod:
  stage: stop
  script:
   - echo "STOP preprod env"
  only:
   - main
  when: manual
  environment:
    name: preprod
    action: stop
```

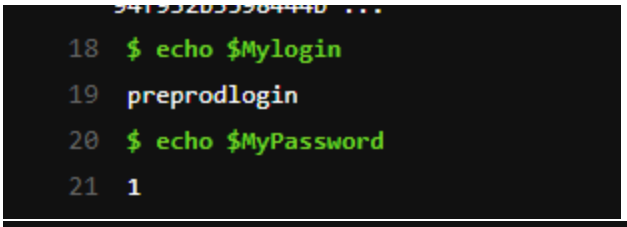
3)

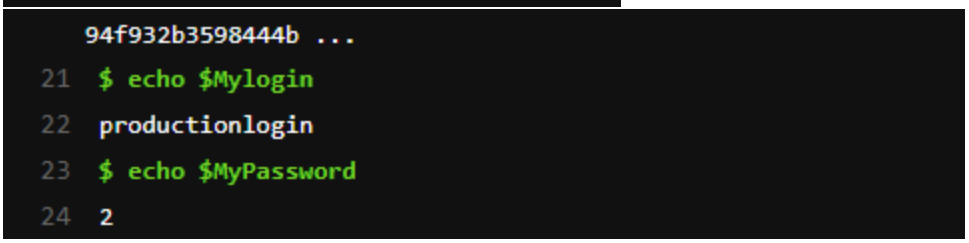
Какого-то красивого (правильного) способа это сделать я не нашел. Поэтому просто добавил глобальную переменную “`Mylogin: preprodlogin`”, а в job production переопределил значение этой переменной “`Mylogin: productionlogin`”.

4)









*)

5.1) мы сначала создаем персональный access token(https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html#creating-a-personal-access-token), чтобы у нас была возможность отменить задание: нажимаем на своего аватара (в верхнем правом угле) -> Setting -> User Settings -> Access Tokens. Даём ему все права.
5.2) Теперь нам нужно сохранить этот персональный access token в переменную $RUNNER_TOKEN: Setting -> CI / CD -> Variables -> Expand -> Add Variable
5.3) Меняем .gitlab-ci.yml.
5.3.1) Что бы нам не скачивать пакеты curl и jq, сразу используем готовый images(everpeace/curl-jq).

5.3.2) Создаём 2 jobs
5.3.2.1) cancel - проверяет и останавливает предыдущие pipelines
5.3.2.1.1) if ["$CI_COMMIT_REF_NAME" != "main"] - мы не хотим чтобы останавливались pipelines на мастре (особенно на prod)
5.3.2.1) curl -s -H "PRIVATE-TOKEN: $RUNNER_TOKEN" "https://gitlab.com/api/v4/projects/${CI_PROJECT_ID}/pipelines?ref=${CI_COMMIT_REF_NAME}&status=running" - получаем список работающих pipelines на текущей ветки.
5.3.2.1) curl -s --request POST -H "PRIVATE-TOKEN: ${RUNNER_TOKEN}" "https://gitlab.com/api/v4/projects/${CI_PROJECT_ID}/pipelines/${pipeline}/cancel - останавливаем pipelines в цикле по списку.
5.3.2.1) || echo "Canceling old pipelines (${OLD_PIPELINES}) failed" - предупреждаем, если что-то пошло не так.
5.3.2.2) sleep - просто ждёт 900 секунд
5.3.) проверяем на ветки main. Не должны отменяться предыдущие.
5.3.) проверяем на ветки task_cancel_me(нужно создать новый branch task_cancel_me). Должны отменяться предыдущие.

```
stages:
- stop previous jobs
- new job
cancel:
stage: stop previous jobs
image: everpeace/curl-jq
script:
- |
if [ "$CI_COMMIT_REF_NAME" != "main" ]
then
(
echo "Cancel old pipelines from the same branch except last"
OLD_PIPELINES=$( curl -s -H "PRIVATE-TOKEN: $RUNNER_TOKEN"
"https://gitlab.com/api/v4/projects/${CI_PROJECT_ID}/pipelines?ref=${CI_COMMIT_REF_NAME}&status=running" \
| jq '.[] | .id' | tail -n +2 )
for pipeline in ${OLD_PIPELINES}; \
do echo "Killing ${pipeline}" && \
curl -s --request POST -H "PRIVATE-TOKEN: ${RUNNER_TOKEN}"
"https://gitlab.com/api/v4/projects/${CI_PROJECT_ID}/pipelines/${pipeline}/cancel"; done
) || echo "Canceling old pipelines (${OLD_PIPELINES}) failed"
fi
sleep:
stage: new job
script:
- sleep 900
```