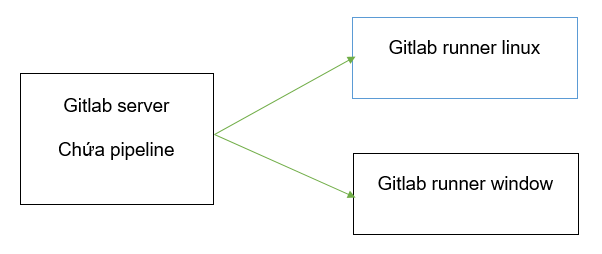
# Install gitlab-runner



[Install GitLab Runner manually on GNU/Linux | GitLab](https://docs.gitlab.com/runner/install/linux-manually.html)

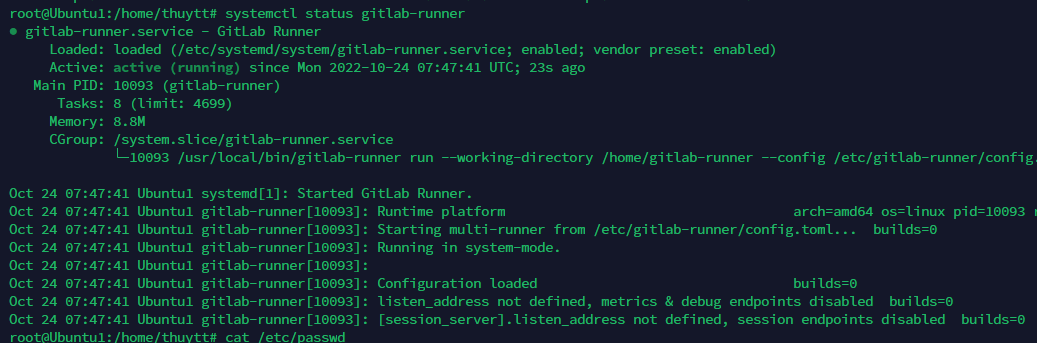
sudo curl -L --output /usr/local/bin/gitlab-runner "https://gitlab-runner-downloads.s3.amazonaws.com/latest/binaries/gitlab-runner-linux-amd64"

sudo chmod +x /usr/local/bin/gitlab-runner

sudo useradd --comment 'GitLab Runner' --create-home gitlab-runner --shell /bin/bash

sudo gitlab-runner install --user=gitlab-runner --working-directory=/home/gitlab-runner

sudo gitlab-runner start



Bước 2: register gitlab runner cho gitlab

Nếu ta ko có server gitlab riêng để lưu trữ pipeline thì ta dùng gitlab.com, gitlab chung để register

Coi gitlab.com là con server đi

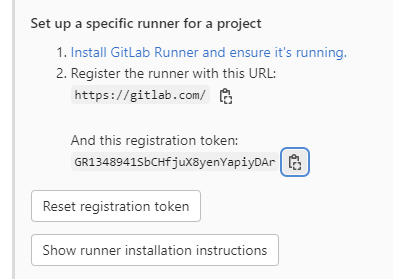
Vào gitlab runner (con được define là gitlab runner)

gitlab-runner register

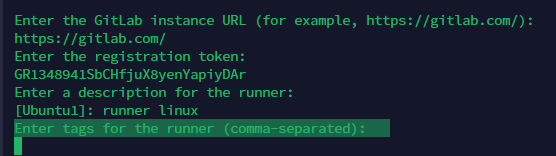


Nó sẽ hỏi URL của bạn là gì,

Vào gitlab -> pipeline -> setting -> CI/CD -> Runners ->

 copy cái 2, sau cái 2 là registration token

Bạn cũng có thể ấn vào show runner installation instructions để có command install gitlab-runner



**tags for the runner (comma-separated)**

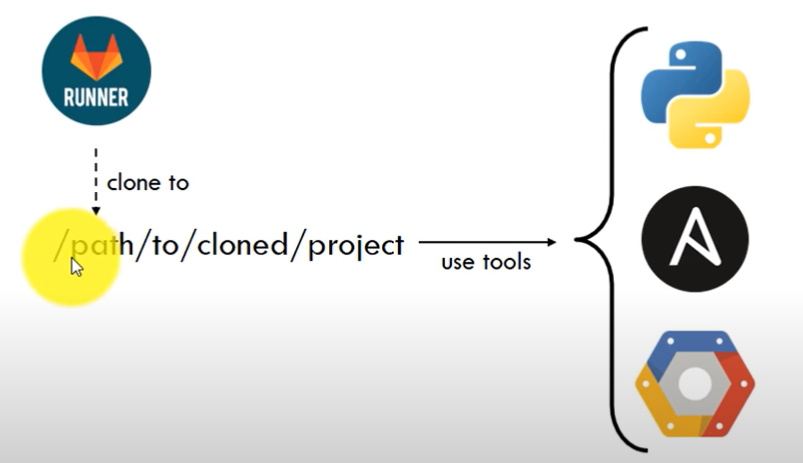
tags quyết định là runner nào sẽ chạy cho project đó, tag runner = deploy host

tôi sẽ để tag đây là linux (nếu ta có 2 runner là linux và ubuntu)

Enter an executor: docker-ssh, shell, ssh, virtualbox, docker+machine, instance, custom, parallels, docker-ssh+machine, kubernetes, docker:

Ta có các loại executor sau đây

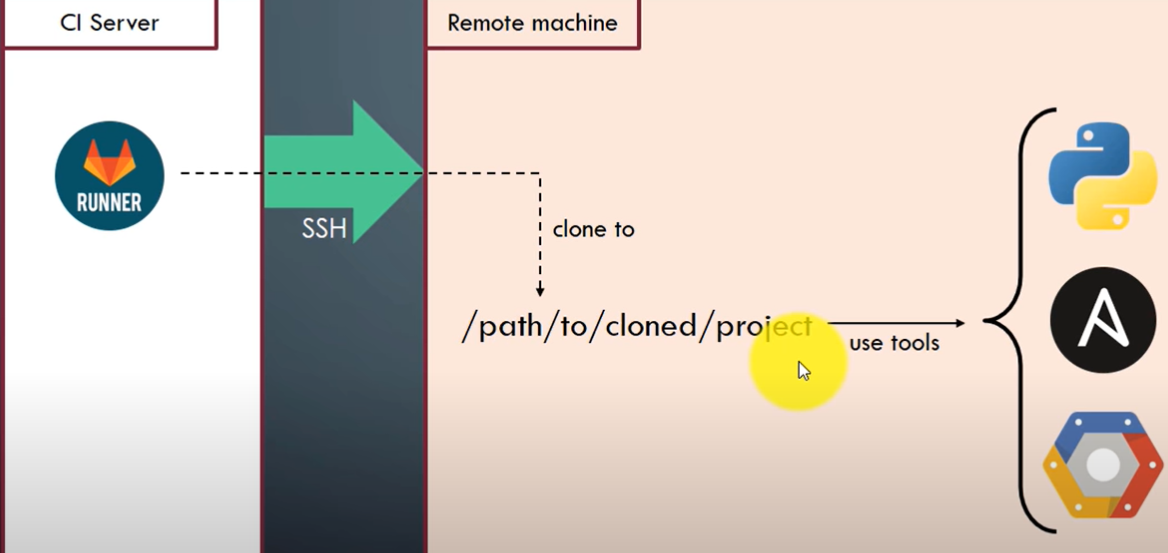
1. Shell là executor đơn giản nhất



Khi chạy thì runner sẽ clone project vào chính CI/CD server, sử dụng luôn môi trường CI/CD server là môi trường runner pipeline, tất cả các tool cần thiết đều phải được cài đặt trước

1. Ssh executor

Cơ bản cũng như shell, tuy nhiên khi chạy thì runner sẽ ssh vào 1 remote machine, sau đấy sử dụng môi trường của remote machine là môi trường run bai bờ lai , các tool cần thiết cũng cần phải đk cài đặt trk



1. Virtualbox executor

Trước khi chạy, runner tạo ra máy ảo

1. Docker executor

Cũng tương tự, runner sẽ tạo ra container, tuy nhiên container sẽ nhẹ hơn rất nhiều so vs máy ảo

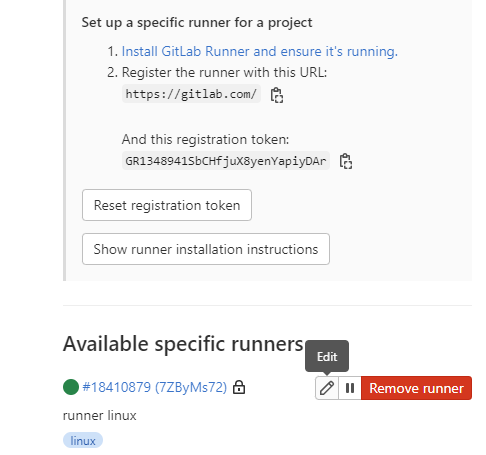
1. Kubernetes executor

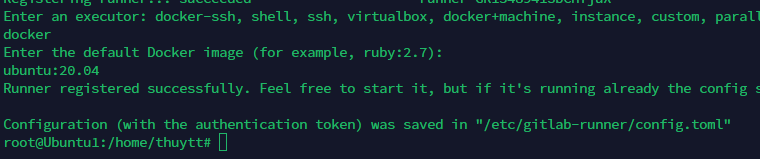
Chắc ko cần phải nói nữa

* Tôi sẽ cài executor là docker

Nhớ cài docker trên máy runner để n có thể tạo runner

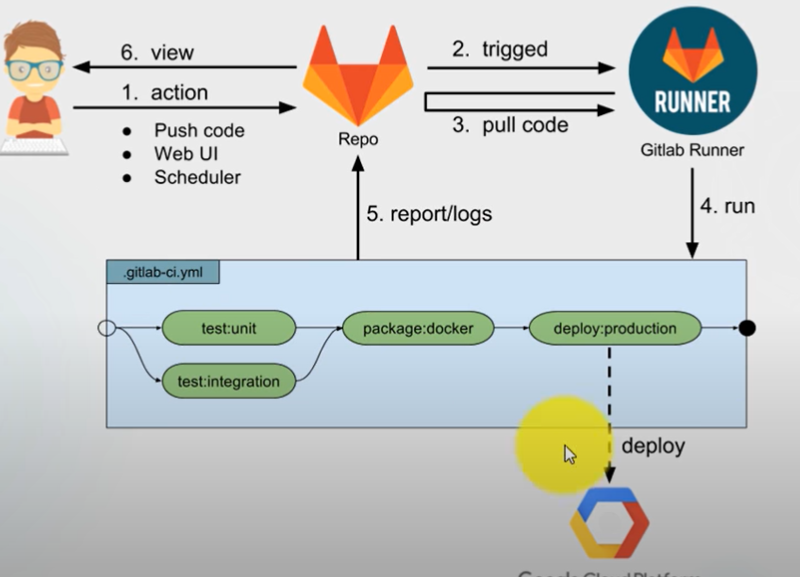
Khi cài đặt xog, refresh lại gitlab ta sẽ thấy 1 runner ở đây

, nếu config sai có thể phi vào sửa, hoặc ta có thể vào **/etc/gitlab-runner/config.toml**



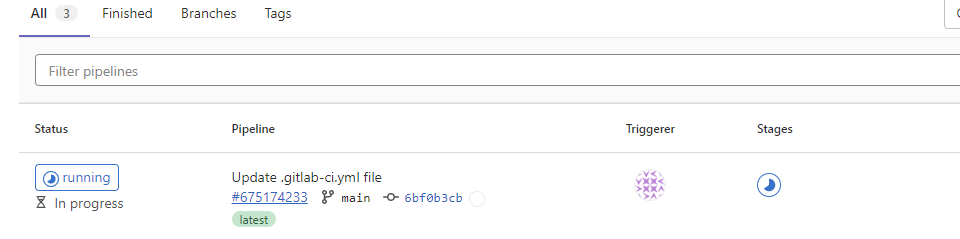
## Chạy 1 job CICD

Cần 1 file tên gitlab-ci.yaml trên chính project, file này định nghĩa các job mà ta muốn thực hiện

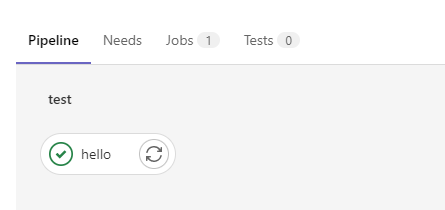
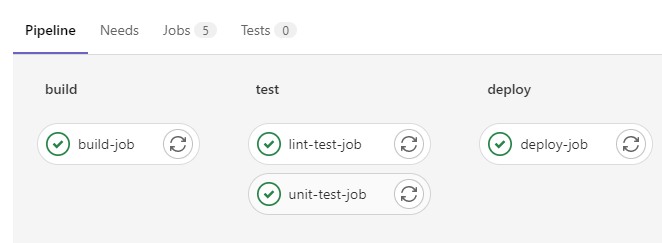


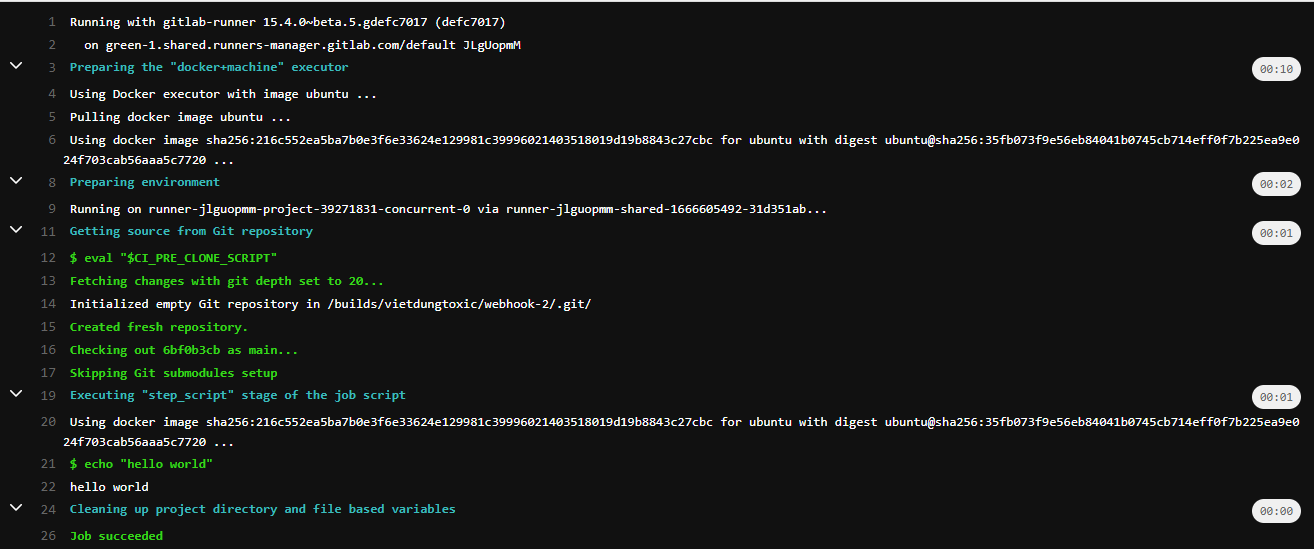
Tạo 1 file .gitlab-ci.yml

Sang CICD -> pipeline để xem kqua, n đang running đây



ấn running sẽ có các stage ở đây, bấm từng stage sẽ



# Gitlab:CI-CD

CI: Continuous Integration

CD:

* Continuous Delivery (chuyển giao liên tục)
* Continuous Deployment (triển khai liên tục).

## Gitlab pipeline

stages: # chỉ định những tag cần chạy

  - test

  - deploy

test1:

  image: ubuntu:20.04

  stage: test             # đánh tag cho stage để gọi

  script:

  - apt update -y

  - mkdir xxx

test2:

  image: electronuserland/builder:wine

  stage: deploy

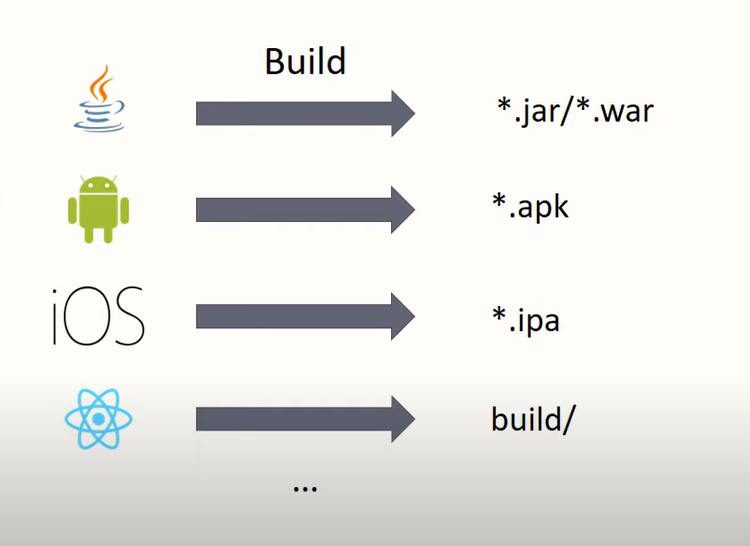
  script:

  - wget https://dl.min.io/client/mc/release/linux-amd64/mc

  - pwd

## Auto build với gitlab CI

Build artifact



Artifact là thành quả của quá trình build

Ví dụ project java khi build xog ta sẽ dk 1 file .jar/.war

Web app là cả folder build

build:react:

  image: build:8.9-alpine

  stage: test             # đánh tag cho stage để gọi

  cache:

    key: npm-cache        # job nào cùng key sẽ set cache cho nhau

    paths: [ .npm ]       # đường dẫn file muốn cache lại

  before\_script:

    - npm config set cache .npm  # set cache cho npm ở folder .npm

    - npm install

  script:

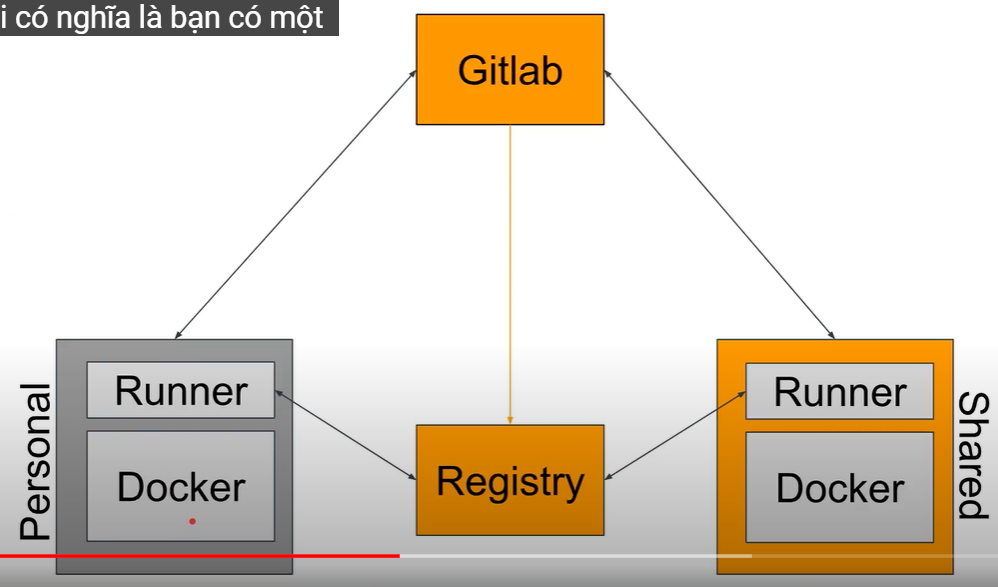
  - npm run build

  artifacts:

    paths:

      - ./build         # folder chứa source build

    expire\_in: "30 days"

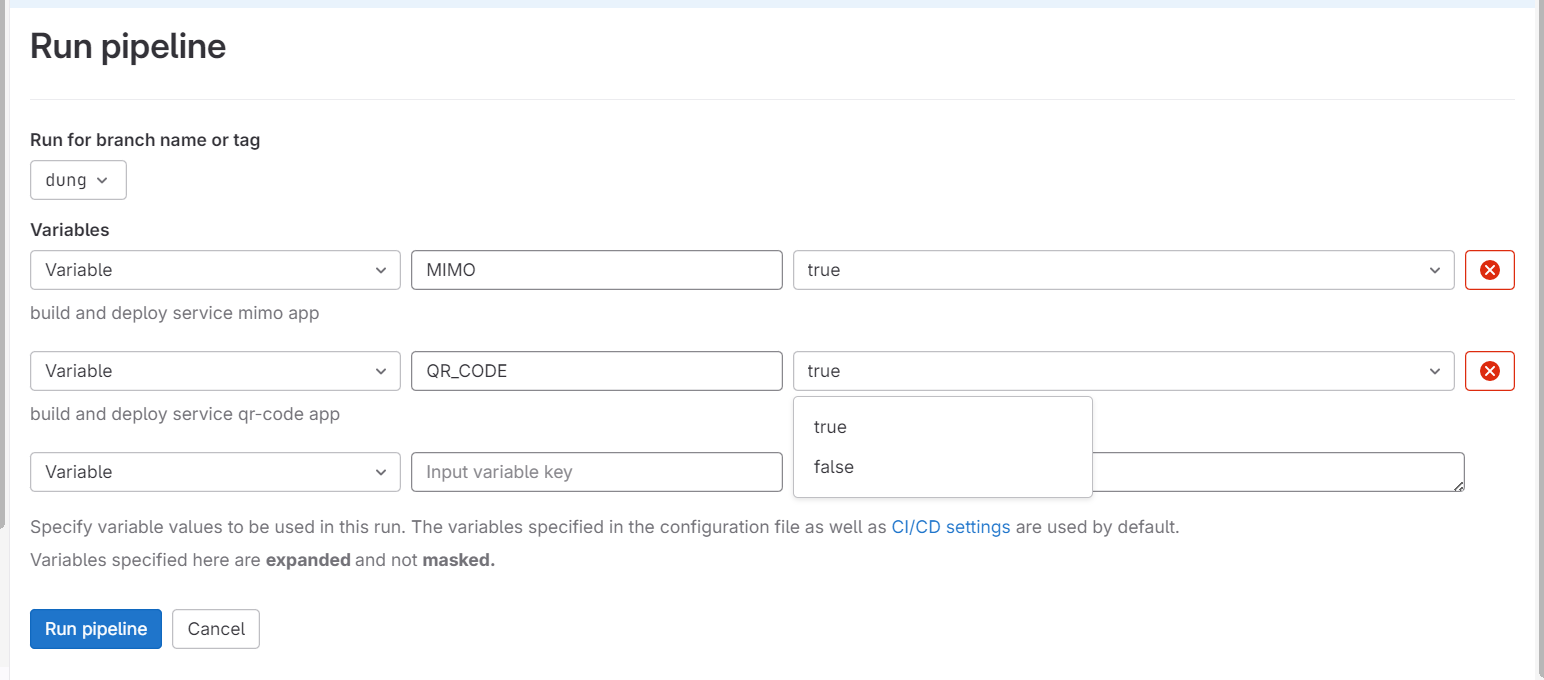


Những cái mày vàng là tài sản của gitlab

Tức là khi script chạy vs image mà ta ko chỉ định tag deploy trên host nào thì n đang chạy trên container của chính máy chủ gitlab

Vì vậy để CI/CD cần add card để chúng n cho thuê máy chủ để chạy task của mình

## CI-CD chọn ENV



Chọn ENV khi run pipeline

variables:

  MIMO:

    value: "true"

    options:

      - "true"

      - "false"

    description: "build and deploy service mimo app"

  QR\_CODE:

    value: "true"

    options:

      - "true"

      - "false"

    description: "build and deploy service qr-code app"

stages:

#  - test

  - build

  - deploy

build image for STAGING:

  stage: build

  variables:

    BUILD\_CONTEXT: .

  extends:

    - .rules:release:staging

  before\_script:

    - docker login -u $STG\_CI\_REGISTRY\_USER -p $STG\_CI\_REGISTRY\_PASSWORD $STG\_CI\_REGISTRY

  script:

    - if [ "$MIMO" = "true" ]; then

    - echo staging1 ; fi

    - if [ "$QR\_CODE" = "true" ]; then

    - echo staging2 ; fi

  interruptible: true

.if-merge-request-pipeline: &if-merge-request-pipeline

  if: $CI\_PIPELINE\_SOURCE == "merge\_request\_event"

.if-merge-request-to-master: &if-merge-request-to-master

  if: $CI\_MERGE\_REQUEST\_TARGET\_BRANCH\_NAME == "develop"

.if-default-branch: &if-default-branch

  if: $CI\_COMMIT\_BRANCH == "develop"

.if-release-to-staging: &if-release-to-staging

  if: $CI\_COMMIT\_TAG && $CI\_COMMIT\_TAG =~ /^[0-9]+\.[0-9]+\.[0-9]+(-rc[0-9]+)$/ && $CI\_PIPELINE\_SOURCE == 'web'

.if-release-to-production: &if-release-to-production

  if: $CI\_COMMIT\_TAG && $CI\_COMMIT\_TAG =~ /^[0-9]+\.[0-9]+\.[0-9]+$/ && $CI\_PIPELINE\_SOURCE == 'web'

.rules:release:development:

  environment:

    name: development

  rules:

  - <<: \*if-merge-request-pipeline

    when: never

  - <<: \*if-default-branch

  - <<: \*if-merge-request-to-master

.rules:release:staging:

  environment:

    name: staging

  rules:

  - <<: \*if-release-to-staging

$CI\_PIPELINE\_SOURCE == 'web' chỉ chạy CI-CD khi bấm run pipeline

Build khi có thay đổi với commit trước

build\_variable:

    stage: build

    # rules:

      # - if: $DEPLOY\_ENVIRONMENT == "staging"

    script:

      - |

        if git diff --name-only $CI\_COMMIT\_SHA^..$CI\_COMMIT\_SHA -- app | grep -q 'app'; then

          echo "1"

        elif git diff --name-only $CI\_COMMIT\_SHA^..$CI\_COMMIT\_SHA -- ci-cd | grep -q 'ci-cd'; then

          echo "2"

        else

          echo "1"

          echo "2"

        fi

# Trong ví dụ này, lệnh git diff được sử dụng để so sánh các thay đổi giữa

# commit trước ($CI\_COMMIT\_SHA^) và commit hiện tại ($CI\_COMMIT\_SHA). Bằng cách sử dụng --name-only,

# chỉ danh sách các tệp tin được trả về. Sau đó, lệnh grep được sử dụng để kiểm tra xem có tệp tin nào

# trong danh sách có đường dẫn bắt đầu bằng app/project/ hay không

build\_variable:

    stage: build

    # rules sẽ áp dụng cho cả script

    # rules:

      # - if: $DEPLOY\_ENVIRONMENT == "staging"

    script:

      - if git diff --name-only $CI\_COMMIT\_SHA^..$CI\_COMMIT\_SHA | grep -q '^app/project/'; then echo "Code changes detected"; fi