GABRIELE ROMANATO

Come creare una pipeline su GitLab

GitLab CI/CD è una potente suite di strumenti integrata in GitLab che permette di automatizzare il processo di build, test e deploy del codice. In questo articolo, vedremo come creare una pipeline CI/CD in GitLab per pubblicare automaticamente le modifiche su un server di produzione ogni volta che viene effettuato un push su una repository.

Prima di iniziare, assicurati di avere i seguenti requisiti:

- Un account GitLab con accesso a un progetto GitLab.
- Un server di produzione con accesso SSH.
- Chiavi SSH configurate per l'accesso senza password al server di produzione.
- Un file di configurazione `.gitlab-ci.yml` nella tua repository GitLab.

Per consentire a GitLab di connettersi al tuo server di produzione, dovrai configurare le chiavi SSH. Genera una chiave SSH sul tuo computer locale (se non ne hai già una):

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

Aggiungi la chiave pubblica al file `~/.ssh/authorized_keys` sul server di produzione.

Aggiungi la chiave privata come variabile segreta su GitLab:

- Vai al tuo progetto GitLab.
- Naviga su **Settings > CI / CD > Variables**.
- Aggiungi una nuova variabile con il nome `SSH_PRIVATE_KEY` e incolla il contenuto della tua chiave privata.

Il file `.gitlab-ci.yml` definisce le pipeline CI/CD. Ecco un esempio di configurazione base per il deploy su un server di produzione:

```
stages:
  - build
  - deploy

variables:
  SSH_KEY: $SSH_PRIVATE_KEY
  DEPLOY_SERVER: "user@your_production_server"
  DEPLOY_PATH: "/path/to/your/project"

before_script:
  - 'which ssh-agent || ( apt-get update -y && apt-get
install openssh-client -y )'
  - eval $(ssh-agent -s)
  - echo "$SSH_KEY" | tr -d '\r' | ssh-add -
  - mkdir -p ~/.ssh
  - chmod 700 ~/.ssh
  - ssh-keyscan -H $DEPLOY_SERVER >> ~/.ssh/known_hosts
  - chmod 644 ~/.ssh/known_hosts

build:
  stage: build
  script:
    - echo "This is where you would put build commands, e.g.,
npm install"

deploy:
  stage: deploy
  script:
    - echo "Deploying to production server..."
    - rsync -avz --delete-after --exclude '.git' ./
$DEPLOY_SERVER:$DEPLOY_PATH
  only:
    - master
```

In dettaglio:

- **Stages**: Definisce le fasi della pipeline. In questo esempio, abbiamo due fasi: `build` e `deploy`.
- **Variables**: Definisce le variabili d'ambiente. Usiamo `SSH_PRIVATE_KEY`, `DEPLOY_SERVER` e `DEPLOY_PATH`.
- **before_script**: Contiene i comandi da eseguire prima dei job di ogni stage. Qui configuriamo l'ambiente SSH.
- **build**: Un esempio di job di build. Puoi aggiungere i comandi necessari per costruire il tuo progetto.
- **deploy**: Il job di deploy che utilizza `rsync` per copiare i file sul server di produzione. Questo job viene eseguito solo sui push sul branch `master`.

Dopo aver configurato il file `.gitlab-ci.yml`, effettua un commit e un push sulla branch `master`. GitLab inizierà automaticamente la pipeline CI/CD. Puoi monitorare l'avanzamento e vedere i log di ogni job nella sezione **CI / CD > Pipelines** del tuo progetto GitLab.

Conclusione

Abbiamo visto come configurare una pipeline CI/CD su GitLab per pubblicare automaticamente le modifiche su un server di produzione. Questa configurazione base può essere estesa e personalizzata per adattarsi alle esigenze specifiche del tuo progetto, includendo ulteriori fasi di test, notifiche e rollback automatici. La chiave per un processo CI/CD di successo è l'automazione e la ripetibilità, assicurando che ogni modifica al codice venga testata e distribuita in modo consistente.