Instalando e Configurando Gitlab CI para Projetos Web

Instalação do Runner

Setup Gitlab Runner

A abordagem usada para o Runner foi de usar um container dentro do nosso servidor de teste com o seguinte comando:

```
docker run -d --privileged --name gitlab-runner /--restart always -v
/var/run/docker.sock:/var/run/docker.sock -v /$PWD/config:/etc/gitlab-runner gitlab/gitlab-
runner:latest
```

Registrando o Runner

Com o container criado, deve-se registrar o runner do seu projeto lá dentro dele com o comando:

```
docker exec -it gitlab-runner gitlab-runner register /-n --url
https://gitlab.dev.evologica.com.br/ --registration-token <TOKEN_DO_PROJETO> /--executor
docker --description "<NOME_DO_PROJETO> CI" --docker-image "docker:latest" --docker-privileged
```

onde `TOKEN_DO_PROJETO` é o token que se encontra na pagina de configuração do CI/CD do projeto, e `NOME_DO_PROJETO` é o nome do projeto para identificar o container lá dentro.

Setup do gitlab-ci.yml

Na raiz do projeto fica o arquivo de configuração, que determina como o CI/CD vai funcionar. Exemplo

do arquivo `yml` do ucHealth:

```
stages:
  - build
  - test
  - publish
  - deploy
build:
  image: node:8-slim
  stage: build
  cache:
    key: node_modules
    paths:
      - cli/web/node_modules/
  before_script:    - yarn config set registry
http://nexus.dev.evologica.com.br/repository/npm/
  script:
    - cd cli/web
    - yarn
    - yarn run build --progress=false
  artifacts:
    paths:
      - cli/web/dist/
  tags:
    - node
test:
  image: node:8-slim
  stage: test
  cache:
    key: node_modules
    paths:
      - cli/web/node_modules/
  script:
    - cd cli/web
    - yarn run lint
  tags:
    - node
```

```
publish:
  image: docker:latest
  stage: publish
  services:
    - docker:dind
  only:
    - "master"
  before_script:    - docker login -u gitlab-ci-token -p $CI_BUILD_TOKEN
registry.dev.evologica.com.br
  script:
    - cd cli/web    - "PACKAGE_VERSION=$(cat package.json | grep version | head -1 | awk -F: '{
print $2 }' | sed 's/[\",]//g' | tr -d '[[:space:]]')"    - docker build -t
registry.dev.evologica.com.br/curio/uc-health:$PACKAGE_VERSION .    - docker push
registry.dev.evologica.com.br/curio/uc-health:$PACKAGE_VERSION    - printf
"VERSION=${PACKAGE_VERSION}\n HOST_URL=test.evologica.com.br" > .env
  after_script:
    - docker logout registry.dev.evologica.com.br
  artifacts:
    paths:
      - cli/web/.env
  tags:
    - docker
deploy:
  image: gitlab/dind:latest
  stage: deploy
  only:
    - "master"
  before_script:    - docker login -u gitlab-ci-token -p $CI_BUILD_TOKEN
registry.dev.evologica.com.br
  before_script:
    - mkdir -p ~/.ssh
    - echo "$DEPLOY_SERVER_PRIVATE_KEY" | tr -d '\r' > ~/.ssh/id_rsa    - chmod 600
~/.ssh/id_rsa
    - eval "$(ssh-agent -s)"
    - ssh-add ~/.ssh/id_rsa
    - ssh-keyscan -H test.evologica.com.br >> ~/.ssh/known_hosts
```

```
  script:
    - cd cli/web    - scp -r .env docker-compose.yml admin@test.evologica.com.br:~/services/uc-health
    - ssh admin@test.evologica.com.br "cd ~/services/uc-health; docker-compose up -d"  #when: manual
  tags:
    - docker
```

com isso já deve ser capaz de realizar o CI/CD.

Autenticação

Para realizar Publicação e Deploy, requer um nivel de autenticação.

Personal Access Token

Em `Settings` -> `Secret variables` deve ser adicionada uma nova variável com nome `CI_BUILD_TOKEN` onde seu valor é o seu personal access token. Esse token será exigido em comandos como este: `docker login -u gitlab-ci-token -p $CI_BUILD_TOKEN registry.dev.evologica.com.br`.

Private Key

Outra variável secreta que deve ter para realizar o deploy é a `DEPLOY_SERVER_PRIVATE_KEY`, onde seu valor deve ser a cahve privada para o acesso na maquina onde será realizado do deploy.

Deploy Node Clients

Para realizar o deploy de clientes Nodes, usamos um container web service. Portanto devemos criar o `docker-compose.yml` e o `Dockerfile` para subir esse service na máquina de teste.

docker-compose.yml

exemplo do ucHealth:

```
version: '2'
services:
  ucHealth:
    image: registry.dev.evologica.com.br/curio/uc-health:${VERSION}    container_name:
ucHealth
    volumes:
      - ./dist:/var/www/html
    environment:
      - VIRTUAL_HOST=ucHealth.${HOST_URL}
      - VIRTUAL_PORT=80
      - LETSENCRYPT_HOST=ucHealth.${HOST_URL}
      - LETSENCRYPT_EMAIL=isaac@conexops.com.br
networks:
  default:
    external:
      name: webproxy
```

Dockerfile

exemplo do ucHealth:

```
FROM nginx:alpine
COPY dist/ /usr/share/nginx/html
```

.

Revision #7
Created Mon, Mar 12, 2018 3:01 PM by brunoagrizzi
Updated Thu, Mar 5, 2020 8:52 PM by Erika