CI / CD JOBS AND STAGE

1) JOBS AND STAGES

*) A pipeline can deploy to one or more environments.
*) A stage is a way of organizing jobs in a pipeline and each stage can have one or more jobs.
*) Each job runs on one agent. A job can also be agentless.

THREE TYPE OF STAGES MAINLY USED

(But in infrastructure we can use many stages)

i) Build
ii) Test
iii) Deploy

Create a file this format========= .gitlab-ci.yml

01-02-2024

VARIABLES

Some variables are available when GitLab creates the pipeline, and can be used to configure the pipeline or in job scripts.

Variables it can help to stored values:

Ex : name: vijay

Type of variables:

1)Global variable—----------> last priority

2) jobs variable (local variable)------> second priority

3) pre-defined variable—---------> first priority

Three way to call the variables :

1) Echo “hai am in $name” 2) ‘name’ 3)$()name

1) GLOBAL VARIABLE:

This global variable applicable for all jobs

Step 1 : create yaml file

```
stages:
  - "dev"
  - "test"
  - "deploy"
variables:
   NAME: vijay
jobs_1:
   image: httpd
   stage: dev
   script:
       - echo "my name is $NAME "
```

Step 2: Default image (rubey)-----> we can change the image

```
Using Docker executor with image httpd ...
Pulling docker image httpd ...
Using docker image sha256:2cfd65f8d6ffc26df43dad28298bfd6132b134510c7b1ac
7a2768dfe379689cf for httpd with digest httpd@sha256:ba846154ade27292d21
cce2d21f1c7e589f3b66a4a643bff0cdd348efd17aa3 ...
```

Step 3 : check the output go to (build—>pipline)

```
14 Checking out 9fa12d46 as detached HEAD (ref is main)...
15 Skipping Git submodules setup
16 $ git remote set-url origin "${CI_REPOSITORY_URL}"
17 Executing "step_script" stage of the job script
18 Using docker image sha256:2cfd65f8d6ffc26df43dad28298bfd6
   7a2768dfe379689cf for httpd with digest httpd@sha256:ba84
   cce2d21f1c7e589f3b66a4a643bff0cdd348efd17aa3 ...
19 $ echo "my name is $NAME "
20 my name is vijay
21 Cleaning up project directory and file based variables
```

2) JOBS VARIABLE (LOCAL VARIABLE)

It will stored in locally
—--- Inside the jobs

Step 1 : write the yaml file

```yaml
stages:
  - "dev"
  - "test"
  - "deploy"
variables:
   NAME: vijay
jobs_1:
   variables:
      NAME: appu
   image: httpd
   stage: dev
   script:
       - echo "my name is $NAME "
```

Step 2 : Check the output

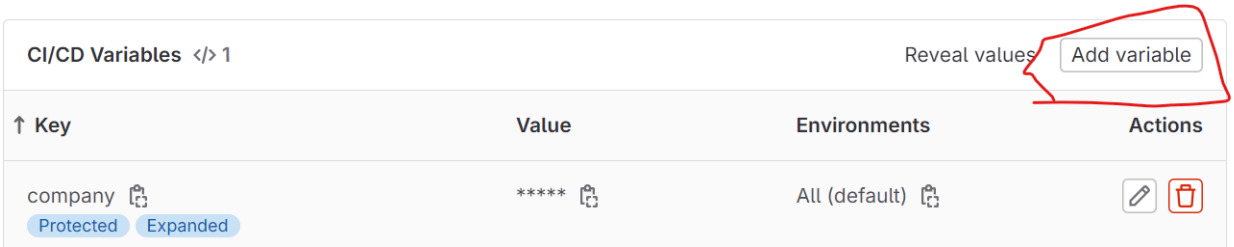
```
$ git remote set-url origin "${CI_REPOSITORY_URL}"
Executing "step_script" stage of the job script
Using docker image sha256:2cfd65f8d6ffc26df43dad28298bfd61
7a2768dfe379689cf for httpd with digest httpd@sha256:ba84
cce2d21f1c7e589f3b66a4a643bff0cdd348efd17aa3 ...
$ echo "my name is $NAME "
my name is appu
Cleaning up project directory and file based variables
```

3) Predefined variable:

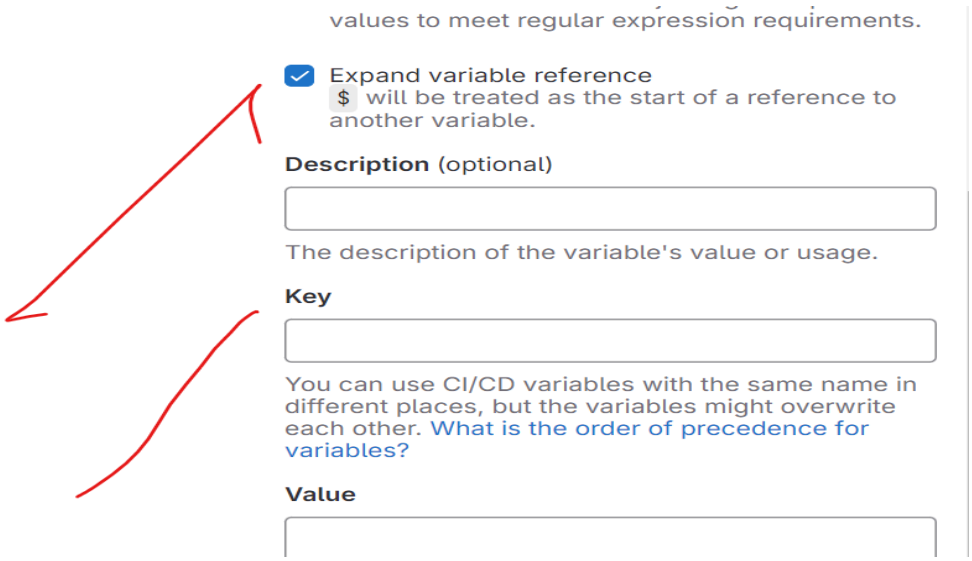
It will pre-defined on ci/cd (in your system)

Path : setting—-> ci/cd—->variables—---->add variables—> create key and value

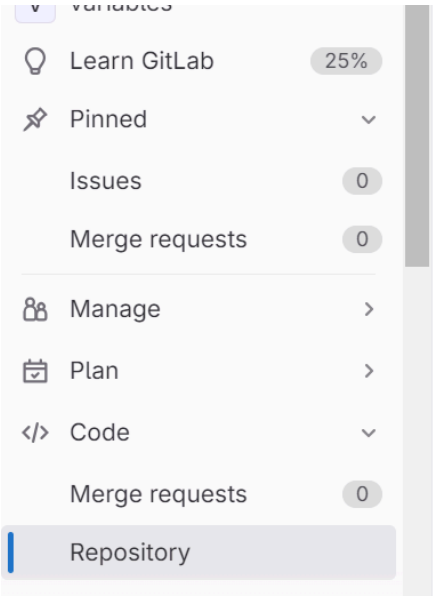
Step 1: create pre=defined variables



Step 2 : create key and value:



Step 3 : Create yaml file



.gitlab-ci.yml 184 B

```yaml
stages:
  - "dev"
  - "test"
  - "deploy"
variables:
   NAME: vijay
jobs_1:
   variables:
      name: appu
   image: httpd
   stage: dev
   script:
       - echo "my name is $company"
```

Step 4 : Check the output

```
Using docker image sha256:2cfd65f8d6ffc26df43dad28298bfd6
7a2768dfe379689cf for httpd with digest httpd@sha256:ba84
cce2d21f1c7e589f3b66a4a643bff0cdd348efd17aa3 ...
$ echo "my name is $company"
my name is tcs
Cleaning up project directory and file based variables
Job succeeded
```

ARTIFACTS:

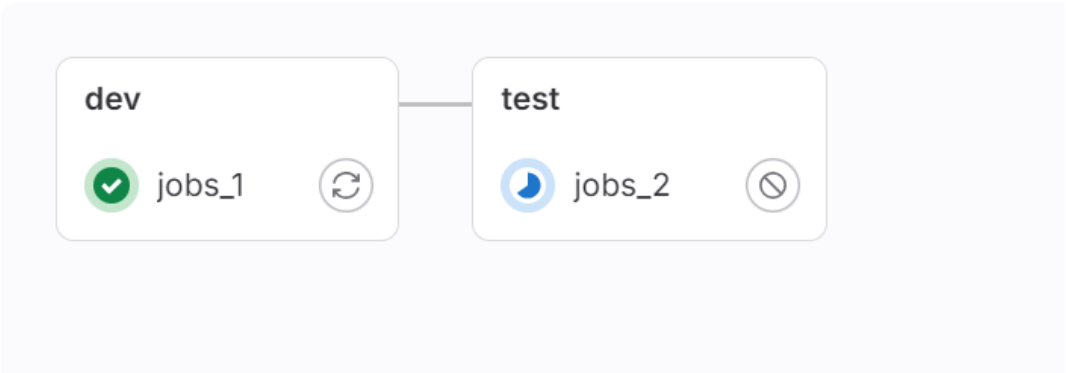
An artifact is a list of files and directories attached to a job after it finishes

Without artifacts we did not connect two are more jobs

Step 1 : create yaml file

```
stages:
  - "dev"
  - "test"
  - "deploy"
variables:
   NAME: vijay
jobs_1:
   image: httpd
   stage: dev
   script:
       - echo "my name is $NAME"
       - mkdir dir
       - echo "simplly waste" > dir/file
       - cat dir/file
   artifacts:
      paths:
         - dir
jobs_2:
    stage: test
    script:
      - cat dir/file
```

Step 2 : check the pipeline



Step 3 : Check the jobs(1)----------> the files are created

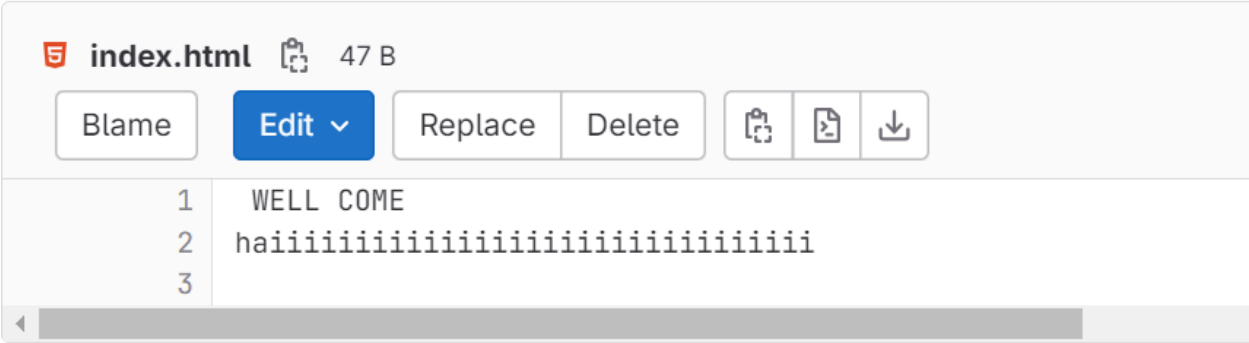
```
cce2d21f1c7e589f3b66a4a643bff0cdd348efd17aa3 ...
$ echo "my name is $NAME"
my name is vijay
$ mkdir dir
$ echo "simplly waste" > dir/file
$ cat dir/file
simplly waste
Uploading artifacts for successful job
Uploading artifacts...
```

Step 4 : Check the jobs(2)---> it will show the output for jobs(1)

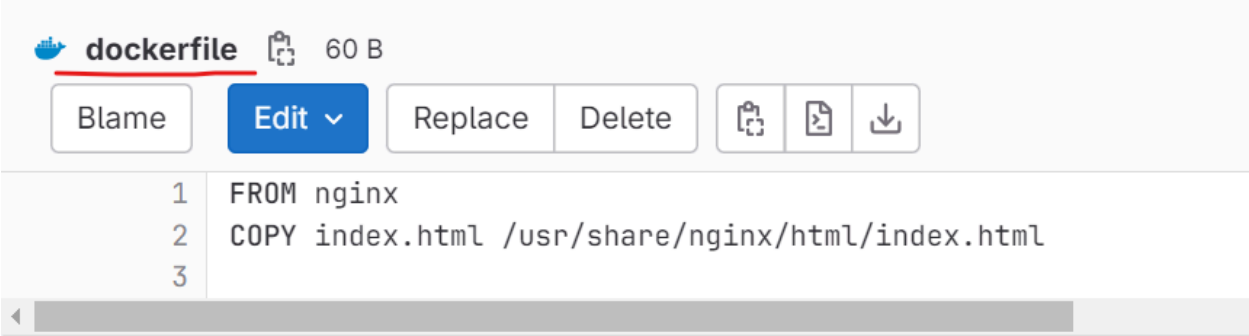
```
Using docker image sha256:ee4548ef50bf8af4f10596e7a12e31
964404280d6f65d0d for ruby:3.1 with digest ruby@sha256:
91ec9321357031879948fdcb08ae5c0d4a505fe8bfb14c ...
$ cat dir/file
simplly waste
Cleaning up project directory and file based variables
```

07-02-2024 GIT LAB —>CICD (BUILD IMAGE)--> PUSH HUB.DOCKER—---> PULL IMAGE IN TO DOCKER

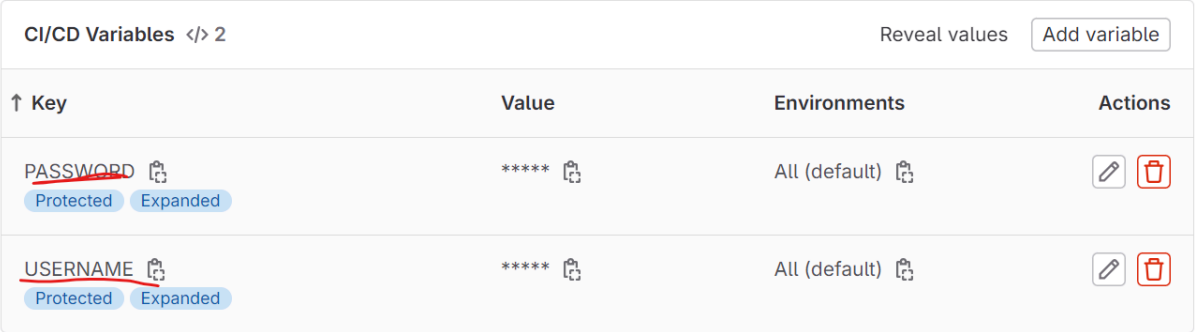
Step 1 : create index.html file in gitlab



Step 2 : Create the docker image in gitlab



Step 3 : Create a variables
Setting —>ci/cd—>variables—-->add variables



Step 4 : Build the docker image in gitlab

```
stages:
  - build

variables:
  DOCKER_IMAGE_TAG: latest

docker-build:
  stage: build
  image: docker:latest
  services:
    - docker:dind
  script:
    - docker --version
    - docker ps
    - docker build -t mynginx:$DOCKER_IMAGE_TAG .
    - echo "$PASSWORD" | docker login -u $USERNAME --password-stdin
    - docker tag mynginx:latest vijayaragavan14012003/dockercicd:latest
    - docker push vijayaragavan14012003/dockercicd:latest
```

Name	Last commit	Last update
.gitlab-ci.yml	Update .gitlab-ci.yml file	5 hours ago
README.md	Initial commit	8 hours ago
dockerfile	Update dockerfile	4 hours ago
index.html	Update index.html	4 hours ago

Step 5 : Give to commit
Step 6 : Check into docker.hub
Step 7 : Next to pull the image and run in your docker platform
Step 8 : Give the ip address and hostport(192.168.255.135:8080) in your browser

08-02-2024

CICD TOMCAT

Github—---->https://github.com/up1/maven_java_web_example/tree/master

Step 1 : Clone the link into your gitbash

```
Admin@DEEPAN-LAPTOP MINGW64 ~ (main)
$ git clone https://github.com/up1/maven_java_web_example.git
Cloning into 'maven_java_web_example'...
remote: Enumerating objects: 167, done.
remote: Counting objects: 100% (47/47), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 167 (delta 30), reused 20 (delta 14), pack-reused 120
Receiving objects:  89% (149/167), 1.30 MiB | 2.46 MiB/s
Receiving objects: 100% (167/167), 5.47 MiB | 5.34 MiB/s, done.
Resolving deltas: 100% (42/42), done.
```

Step 2 : Delete the jenkinsfile

```
Admin@DEEPAN-LAPTOP MINGW64 ~ (main)
$ cd maven_java_web_example/

Admin@DEEPAN-LAPTOP MINGW64 ~/maven_java_web_example (master)
$ ls
Dockerfile   README.md  mvnw*     pom.xml  ui_test/
Jenkinsfile  hello.txt  mvnw.cmd  src/

Admin@DEEPAN-LAPTOP MINGW64 ~/maven_java_web_example (master)
$ cd Jenkinsfile
bash: cd: Jenkinsfile: Not a directory

Admin@DEEPAN-LAPTOP MINGW64 ~/maven_java_web_example (master)
$ rm -r Jenkinsfile

Admin@DEEPAN-LAPTOP MINGW64 ~/maven_java_web_example (master)
$ ls
Dockerfile  README.md  hello.txt  mvnw*  mvnw.cmd  pom.xml  src/  ui_test/
```

Step 3 : git add . and git commit -m “file”

```
Admin@DEEPAN-LAPTOP MINGW64 ~/maven_java_web_example (master)
$ git add .

Admin@DEEPAN-LAPTOP MINGW64 ~/maven_java_web_example (master)
$ git commit -m "tomcat"
[master 3aa3f64] tomcat
 1 file changed, 19 deletions(-)
 delete mode 100644 Jenkinsfile
```

Step 4 : push the file in gitlab

```
Admin@DEEPAN-LAPTOP MINGW64 ~/maven_java_web_example (master)
$ git push https://gitlab.com/vijay6531705/tomcat.git
Enumerating objects: 168, done.
Counting objects: 100% (168/168), done.
Delta compression using up to 8 threads
Compressing objects: 100% (91/91), done.
Writing objects: 100% (168/168), 5.47 MiB | 2.95 MiB/s, done.
Total 168 (delta 42), reused 167 (delta 42), pack-reused 0
remote: Resolving deltas: 100% (42/42), done.
remote:
remote: To create a merge request for master, visit:
remote:   https://gitlab.com/vijay6531705/tomcat/-/merge_requests/new?merge_requ
est%5Bsource_branch%5D=master
remote:
To https://gitlab.com/vijay6531705/tomcat.git
 * [new branch]      master -> master
```

Step 4: Create a repository in gitlab

Step 5 : create yml file

```yaml
stages:
    - build
    - test
build:
    image: maven:latest
    script:
     - mvn -N clean test
     - mvn -N clean package
    artifacts:
     paths:
        - "/builds/vijay6531705/tomcat/target/api.war"
```

Step 6 : commit to change

Step 7: Final result

```
773 Uploading artifacts for successful job                          00:0
774 Uploading artifacts...
775 /builds/vijay6531705/tomcat/target/api.war: found 1 matching artifact fi
    es and directories
776 WARNING: Upload request redirected                  location=https://git
    ab.com/api/v4/jobs/6123312366/artifacts?artifact_format=zip&artifact_typ
    =archive new-url=https://gitlab.com
777 WARNING: Retrying...                                context=artifacts-up
    oader error=request redirected
778 Uploading artifacts as "archive" to coordinator... 201 Created  id=61233
    2366 responseStatus=201 Created token=glcbt-65
779 Cleaning up project directory and file based variables           00:0
```

Step 8 : download to check

Step 9 : extract the api.war file

New:

Step 10 : Create a new repository in docker.hub
Step 11 : come to gitlab to create new repository
step 12 : upload the api.war file in your gitlab
Step 13 : Create docker file in your gitlab

dockerfile 52 B

```
FROM tomcat
COPY api.war /usr/local/tomcat/webapps/

```

Step 14 : Create a variable for USER & PASSWORD

Step 15 : Create yaml file in your gitlab

```yaml
stages:
  - build

docker-build:
  stage: build
  image: docker
  services:
    - docker:dind
  script:
    - docker --version
    - docker build -t tomcat:latest .
    - docker images
    - docker run -d -p 8081:8080  tomcat:latest
    - docker ps
    - docker login -u $USERNAME -p $PASSWORD
    - docker tag tomcat:latest vijayaragavan14012003/tomcat:latest
    - docker push vijayaragavan14012003/tomcat:latest
```

Step 16 : Committee to change and check the output

```
H
$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED                  SIZE
tomcat       latest    511337a438be   Less than a second ago   460MB
$ docker run -d -p 8081:8080  tomcat:latest
9a7a33c4099bb299b001f33ceb7befd92151934244003f8a870e8a0466ab7630
$ docker ps
CONTAINER ID   IMAGE           COMMAND             CREATED
STATUS                  PORTS                                       NAMES
9a7a33c4099b   tomcat:latest   "catalina.sh run"   Less than a second ago
Up Less than a second   0.0.0.0:8081->8080/tcp, :::8081->8080/tcp   happy
_margulis
$ docker login -u $USERNAME -p $PASSWORD
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /root/.docker/confi
g.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-s
tore
Login Succeeded
```

```
eb81a90911ef: Preparing
ab995379f7a6: Preparing
1a102d1cac2b: Preparing
82b56c0ec2cb: Waiting
eb81a90911ef: Waiting
ab995379f7a6: Waiting
1a102d1cac2b: Waiting
44514f573ec0: Layer already exists
547fa6fe2dfd: Layer already exists
82b56c0ec2cb: Layer already exists
eb81a90911ef: Layer already exists
eef0b873181f: Mounted from library/tomcat
ab995379f7a6: Layer already exists
b5471a7413cd: Mounted from library/tomcat
1a102d1cac2b: Layer already exists
b683e11f76ec: Pushed
latest: digest: sha256:b5a9c0ead5c2af706d864faad5b155401893
340156164bf6ba size: 2206
Cleaning up project directory and file based variables
Job succeeded
```

Pull the private repository in docker.hub

Step 1 : Create a repository in docker.hub
Step 2 : In private
Step 3 : Create image in docker
Step 4 : Push the image into docker.hub
First login
Name:
Password:
Step 4: docker push (repository name)
Step 5 :create secrete key

```
kubectl create secret docker-registry group \
--docker-server=docker.io \
--docker-username=vijayaragavan14012003 \
 --docker-password=VIJAYP248 \
 --docker-email=pvijayaragavan2003@gmail.com
```

Step 5 : Create yaml file

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod-httpd
spec:
  containers:
  - name: mycontainer
    image: vijayaragavan14012003/tomcat:latest
    ports:
    - containerPort: 80
      hostPort: 8080
  imagePullSecrets:
  - name: group
~
~
~
~
```

Step 6 : Create po (kubectl create -f pod.yaml)
Step 7 : Get the pod (kubectl get po)