Journal of Applied Multimedia and Networking (JAMN)
Vol.1, No.1, April 2024, pp. xx~xx
e-ISSN: 2548-6853

IMPLEMENTASI CI/CD DENGAN GITLAB DI PSTEAM

CI/CD Implementation With Gitlab In PSTeam

Mohamad Arya Saputra[1], Cahya Miranto, S.S.T., M.Tr.Kom.[2]
[1.2.] Teknik Informatika, Politeknik Negeri Batam
Email: [1]arya95514@gmail.com , [2]cahya@polibatam.ac.id

Info Artikel

Riwayat Artikel:

Received Apr 25th, 2024
Revised Apr 25th, 2024
Accepted Apr 25th, 2024

Keyword:

Continuous Integration
Continuous Deployment
Pipeline
Gitlab
Metode PPDIOO.

ABSTRAK

Penggunaan teknologi informasi dan komunikasi yang semakin pesat menuntut pengembangan perangkat lunak yang kompleks. Salah satu platform repositori yang populer adalah GitLab, yang menawarkan lingkungan kolaboratif untuk pengelolaan kode proyek. Namun, kekurangan fitur Continuous Integration/Continuous Deployment (CI/CD) di GitLab PSTeam, sebuah platform repositori yang digunakan oleh PSTeam di Politeknik Negeri Batam, dapat menghambat proses pengembangan perangkat lunak. Oleh karena itu, laporan ini membahas implementasi sistem CI/CD di GitLab PSTeam dengan tujuan mempersingkat waktu pengembangan perangkat lunak dalam lingkup PSTeam. Metode yang digunakan dalam mengembangkan sistem ini adalah metode PPDIOO, Setelah dilakukan pengujian pada *Pipeline* CI/CD di Gitlab PSTeam yang sudah di konfigurasi, *Pipeline* dapat berjalan sesuai dengan yang diharapkan tanpa ada kendala yang berarti dan dapat membantu dalam mempersingkat waktu pengembangan perangkat lunak dalam lingkup PSTeam.

Corresponding Author:

Mohamad Arya Saputra,
Teknik Informatika,
Politeknik Negeri Batam,
Jl. Ahmad Yani, Tlk. Tering, Kec. Batam Kota, Kota Batam, Kepulauan Riau 29461
Email: arya95514@gmail.com

1. PENDAHULUAN

1.1. Latar Belakang

Berkembang pesatnya teknologi informasi dan komunikasi yang semakin bersaing, penggunaan repositori sangat dibutuhkan [9]. Salah satu platform repositori yang dapat membantu pengembangan perangkat lunak adalah GitLab, sebagai platform repositori kode sumber, menyediakan lingkungan kolaboratif yang dapat digunakan untuk mengelola dan menyimpan perubahan pada kode proyek secara terstruktur.

GitLab itu sendiri pada dasarnya adalah aplikasi alur kerja DevOps berbasis web yang menawarkan berbagai opsi untuk mengelola Continuous Integration dan Continuous Deployment (CI/CD), sambil memberikan pelanggan manfaat dari alat kontrol versi standar [8]. GitLab dan layanannya secara luas digunakan sepanjang penelitian ini untuk berbagai tujuan yang akan dibahas dalam bagian selanjutnya. GitLab juga menyediakan alat manajemen tim dan alur kerja seperti tanggung jawab, tonggak, isu, dan lain-lain, yang dirancang untuk memfasilitasi komunikasi dan kolaborasi dalam tim sepanjang siklus pengembangan [8].

PSTeam (Polibatam Software Team) adalah divisi yang bertanggung jawab dalam pengembangan perangkat lunak di Polibatam. Dalam mengembangkan perangkat lunak, PSTeam membutuhkan platform repositori yang baik. Untuk saat ini, website yang digunakan sebagai platform repositori PSTeam adalah GitLab PSTeam, namun kekurangan fitur CI/CD di GitLab PSTeam dapat menjadi hambatan dalam proses pengembangan perangkat lunak dikarenakan proses push dan debugging pada repositori project di GitLab PSTeam akan memakan waktu lebih lama karena mendeteksi error yang ada pada *source code* dilakukan secara manual.

Oleh karena itu, solusi terbaik dari masalah ini adalah dengan diimplementasikannya sistem CI/CD di GitLab PSTeam, yang dapat melakukan proses mendeteksi error secara otomatis setiap dilakukannya push pada repositori project, dengan adanya *Pipeline* CI/CD ini diharapkan dapat mempersingkat waktu pengembangan perangkat lunak membantu pengembang dapat fokus ke hal yang lebih krusial daripada mendeteksi error secara manual.

1.2. Tinjauan Pustaka

dalam penelitian [7] yang berjudul "Design and Validation of a Scheme of Infrastructure of Servers, under the PPDIOO Methodology, in the University Institution - ITSA" menggunakan metode PPDIOO (Prepare, Plan, Design, Implement, Operate, Optimize). Menurut Leonel Hernandez dan Genett Jimenez, Fokus dari metodologi PPDIOO adalah untuk menentukan aktivitas minimum yang diperlukan, disesuaikan berdasarkan teknologi dan kompleksitas jaringan, agar dapat memberikan saran terbaik kepada entitas dalam pemasangan dan pengoperasian teknologi Cisco dengan sukses. Demikian pula, dapat dioptimalkan kinerjanya sepanjang siklus hidup jaringan.

Dalam Penelitian [2] yang berjudul "Implementasi Continuous Integration dan Continious Deployment Pada Aplikasi Learning Management System di PT. Millennia Solusi Informatika" menyatakan bahwa penerapan praktik CI/CD telah meningkatkan efisiensi proyek. Dengan menerapkan praktik CI/CD, fitur baru aplikasi disebarkan kepada pengguna di setiap pengiriman sprint yang akan selalu melewati tahap uji coba sebelum dikirim ke pengguna. Otomatisasi telah memungkinkan untuk melakukan uji coba yang lebih efisien, sehingga mempercepat proses pengembangan serta menghasilkan aplikasi yang lebih Handal.

Dalam Penelitian [8] yang berjudul "Continuous Integration Using Gitlab" menyatakan bahwa Continuous Integration atau CI adalah praktik pengembangan yang bertujuan untuk menciptakan kode berkualitas tinggi dengan risiko atau ketidakpastian perilakunya yang rendah. Proses ini melibatkan tim pengembang, di mana setiap individu menggabungkan kode mereka ke dalam cabang kode yang saat ini stabil dan telah diuji dengan baik, dikenal sebagai main Line. Selain itu Aspek otomatisasi dari CI juga relevan untuk Continuous Deployment (CD) yang terakhir memastikan bahwa semua proses CI kemudian siap untuk diimplementasikan ke produksi. Otomatisasi pembangunan bersama dengan pengujian memastikan bahwa jumlah langkah yang diperlukan untuk implementasi sangat berkurang dengan menciptakan kode yang stabil, fungsional, dan praktis bebas dari kesalahan.

Menurut [6] dalam penelitiannya yang berjudul "Development of an ERP with CI/CD, Authentication and System Audit", sysbox adalah utilitas manajemen dan eksekusi kontainer sumber terbuka yang mendorong batas-batas teknologi kontainerisasi tradisional, Aspek mendasar yang membedakan Sysbox dari runtime kontainer tradisional adalah kesesuaiannya untuk mengatur kontainer sistem.

2. METODE IMPLEMENTASI

Implementasi ini dilakukan dengan menggunakan metode PPDIOO yaitu singkatan dari Prepare, Plan, Design, Implement, Operate dan Optimize, PPDIOO sendiri adalah metode yang digunakan oleh cisco yang umumnya digunakan pada penelitian jaringan.

Gambar 1. Alur PPDIOO

Prepare atau persiapan, pada tahap ini dilakukannya perencanaan kerja yang baik segi teknologi yang dibutuhkan untuk proses implementasi CI/CD ke depannya, Tahap ini dimulai dari menyiapkan Virtual Box

untuk menyediakan lingkungan Ubuntu yang nantinya digunakan untuk menjalankan *source code* GitLab dan proses implementasi CI/CD pada GitLab tersebut. Pada tahap ini juga dilakukan pendalaman materi mengenai konsep CI/CD dan cara untuk melakukan implementasinya beserta pemahaman mengenai CI *Pipeline*, GitLab Runner dan Sysbox yang nantinya akan menjadi fondasi utama dalam proses implementasi CI/CD pada GitLab PSTeam.

Plan atau perencanaan, pada tahap ini dilakukannya penentuan perangkat lunak yang akan digunakan dalam untuk melakukan proses implementasi CI/CD pada GitLab PSTeam, perangkat lunak yang akan digunakan antara lain Virtual Box, sistem operasi Ubuntu, Gitlab, Gitlab-Runner, serta pengamanan lebih lanjut pada Alur Pipeline menggunakan Sysbox.

Design atau desain, pada tahap ini dilakukan proses konfigurasi pada hal-hal yang dibutuhkan untuk implementasi CI/CD, seperti pembuatan GitLab Runner dan instalasi Sysbox ke CI *Pipeline*, dimulai dengan desain arsitektur CI/CD yang menggunakan PC User sebagai pengguna, melakukan akses kepada website Gitlab PSTeam, proyek yang dibuka oleh user dapat dilakukan trigger untuk menjalankan proses CI/CD yang terdiri dari 3 tahap dasar, Build, Test dan Deploy, selanjutnya desain *Pipeline* CI/CD yang diawali dengan pengguna yang melakukan commit kepada repositori, lalu melakukan trigger proses CI/CD, yang terdiri dari 5 tahap yaitu build, notify build, test, notify test dan deploy.

Implement atau implementasi, pada tahap implementasi ini, hasil dari tahap persiapan, perencanaan dan desain mulai di implementasikan pada GitLab PSTeam, Proses implementasi dari 3 tahap utama, tahap instalasi, tahap registrasi dan tahap konfigurasi, tahap instalasi terdiri dari 2 langkah yaitu instalasi ubuntu dan insalasi gitlab-runner, serta tahap konfigurasi yang juga terdiri dari 2 langkah yaitu konfigurasi runner dan konfigurasi sysbox.

Operate atau pengoperasian, Setelah berhasil mengimplementasikan proses persiapan, perencanaan, dan desain, langkah selanjutnya adalah memulai proses pengoperasian, di mana hasil implementasi sebelumnya dijalankan untuk melakukan pengujian secara manual mulai dari runner, konfigurasi runner, konfigurasi Alur Pipeline, serta *Pipeline* itu sendiri.

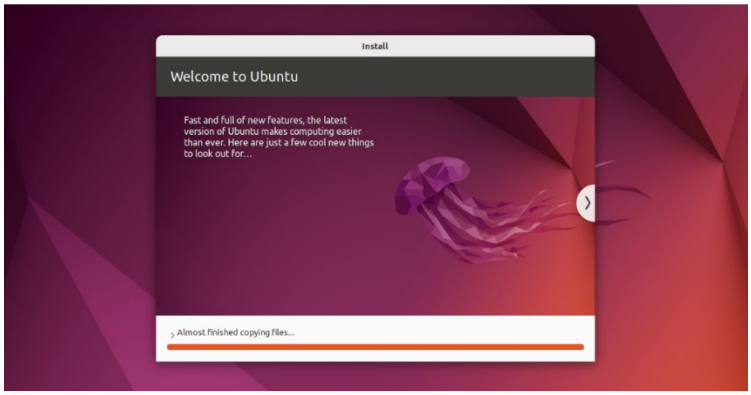
Optimize atau optimasi, adalah tahap terakhir di mana proses CI/CD yang sudah berhasil dioperasikan dimonitor dievaluasi kembali untuk melihat kinerjanya dan melakukan identifikasi potensi masalah atau kebutuhan perubahan yang dapat dilakukan perbaikan atau peningkatan ke depannya, dalam kasus ini akan direvisi Kembali standar konfigurasi Alur Pipeline pada gitlab-ci.yml, agar adanya konfigurasi Alur Pipeline yang sederhana dan kompleks sebelumnantinya pengguna dapat melakukan kustomisasi lebih lanjut sesuai kebutuhan proyek.

3. HASIL IMPLEMENTASI DAN PEMBAHASAN

3.1. Implementasi

3.1.1. Instalasi Ubuntu

Proses implementasi dimulai dengan dilakukan instalasi sistem operasi Ubuntu pada Virtual Box, sistem operasi ini akan menjadi basis utama dalam proses implementasi CI/CD pada Gitlab PSTeam, versi ubuntu yang digunakan pada virtual box ini adalah Ubuntu 22.04.4 LTS, Ubuntu dipilih dikarenakan adanya package manager dan environment terminal yang dapat mempermudah proses instalasi dependencies dan Gitlab-runner itu sendiri



Gambar 2. Instalasi Ubuntu Pada Virtual Box

3.1.2. Instalasi Gitlab-Runner

Pada sistem operasi Ubuntu yang sudah diinstall sebelumnya akan dilakukan proses instalasi Gitlab-Runner serta dependenciesnya pada Website Gitlab PSTeam menggunakan terminal Ubuntu, Gitlab-

Runner nantinya akan berfungsi sebagai platform yang menampung dan membuat runner yang bisa dipakai pada *Pipeline* CI/CD kedepannya.

```
gitlab@gitlab-VirtualBox:~$ sudo apt-get install gitlab-runner
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb
  git-cvs git-mediawiki git-svn docker-engine
The following NEW packages will be installed:
  git git-man gitlab-runner liberror-perl
0 upgraded, 4 newly installed, 0 to remove and 96 not upgraded.
Need to get 497 MB of archives.
After this operation, 558 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://id.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl all 0.1
7029-1 [26,5 kB]
Get:2 http://id.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1
:2.34.1-1ubuntu1.10 [954 kB]
Get:4 http://id.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2
.34.1-1ubuntu1.10 [3.166 kB]
Get:3 https://packages.gitlab.com/runner/gitlab-runner/ubuntu jammy/main amd64 g
itlab-runner amd64 16.11.0-1 [493 MB]
20% [3 gitlab-runner 24,0 MB/493 MB 5%]                    1.723 kB/s 4min 32sS
```

Gambar 3. Instalasi Gitlab-Runner

3.1.3. Registrasi Runner

Setelah instalasi Gitlab-Runner selesai, maka selanjutnya adalah proses registrasi Runner menggunakan perintah "gitlab-runner register" pada terminal Ubuntu, Runner nantinya akan digunakan sebagai pondasi utama Alur Pipeline CI/CD, Runner yang sudah di buat dan di registrasi dapat digunakan untuk menjalankan semua *Pipeline* repositori yang membutuhkannya, selain itu Runner juga dapat dibuat hanya untuk satu *Pipeline* repositori tertentu.

```
psteam@psteam-VirtualBox:~$ gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=5662
 revision=c72a09b6 version=16.8.0
WARNING: Running in user-mode.
WARNING: The user-mode requires you to manually start builds processing:
WARNING: $ gitlab-runner run
WARNING: Use sudo for system-mode:
WARNING: $ sudo gitlab-runner...
```

Gambar 4. Registrasi Runner

Setelah memasukan perintah "gitlab-runner register" pada terminal Ubuntu maka selanjutnya adalah dengan memasukan URL Gitlab PSTeam serta token registrasi yang sudah di generate sebelumnya, dan memilih *executor* untuk runner tersebut menjalankan *Pipeline* nantinya (umumnya shell atau docker).

```
Enter the GitLab instance URL (for example, https://gitlab.com/):
http://10.170.14.86
Enter the registration token:
glrt-5zLXUCBFL_WGFyymdV9B
Verifying runner... is valid                        runner=5zLXUCBFL
Enter a name for the runner. This is stored only in the local config.toml file:
[psteam-VirtualBox]: test
Enter an executor: parallels, docker-windows, kubernetes, docker-autoscaler, ins
tance, custom, shell, ssh, virtualbox, docker, docker+machine:
docker-windows
Enter the default Docker image (for example, mcr.microsoft.com/windows/servercor
e:1809):
mcr.microsoft.com/windows/servercore:1809
Runner registered successfully. Feel free to start it, but if it's running alrea
dy the config should be automatically reloaded!

Configuration (with the authentication token) was saved in "/home/psteam/.gitlab
-runner/config.toml"
```

Gambar 5. Memasukan data runner

Setelah proses registrasi selesai, dijalankan perintah "gitlab-runner run" pada terminal untuk menjalankan runner yang sudah di registrasi sebelumnya, setelah ini Runner yang sudah di registrasi sudah bisa dioperasikan atau bisa dilakukan konfigurasi terlebih dahulu.

```
psteam@psteam-VirtualBox:~$ gitlab-runner run
Runtime platform                                    arch=amd64 os=linux pid=5719
 revision=c72a09b6 version=16.8.0
Starting multi-runner from /home/psteam/.gitlab-runner/config.toml...  builds=0
max_builds=0
WARNING: Running in user-mode.
WARNING: Use sudo for system-mode:
WARNING: $ sudo gitlab-runner...
```

Gambar 6. Memulai runner

3.1.4. Konfigurasi Runner

Setelah runner diregistrasi dan berjalan, selanjutnya adalah melakukan konfigurasi pada runner tersebut pada config.toml menggunakan perintah "sudo nano /etc/gitlab-runner/config.toml", konfigurasi dapat dilakukan dengan memberi nama, URL Gitlab, token registrasi, *executor* dari runner tersebut, jika runner menggunakan docker sebagai *executor* maka ada konfigurasi tambahan untuk docker tersebut seperti image yang digunakan, serta *extra host*.

```
[[runners]]
  name = "test"
  url = "http://10.170.14.86"
  id = 5
  token = "glrt-5zLXUCBFL_WGFyymdV9B"
  token_obtained_at = 2024-04-24T09:20:32Z
  token_expires_at = 0001-01-01T00:00:00Z
  executor = "docker"
  [runners.docker]
    tls_verify = false
    image = "ubuntu:latest"
    privileged = false
    disable_entrypoint_overwrite = false
    oom_kill_disable = false
    disable_cache = false
    extra_hosts = ["docker:10.170.14.86"]
    shm_size = 0
    network_mtu = 0
```

Gambar 7. Konfigurasi Runner pada Config.toml

3.1.5. Konfigurasi Sysbox

Selanjutnya adalah dengan menambahkan konfigurasi sysbox pada runner tersebut pada Config.toml, dengan membuat kolom runner baru yaitu [runners.custom], dan menambahkan eksekusi konfigurasi sysbox, sysbox nantinya akan berperan menjadi lapisan perlindungan tambahan pada isolasi environment yang akan digunakan untuk menjalankan proses build dan test.

```
[runners.custom]
  config_exec = "/usr/bin/sysbox-runc"
```

Gambar 8. Konfigurasi Sysbox pada Config.toml

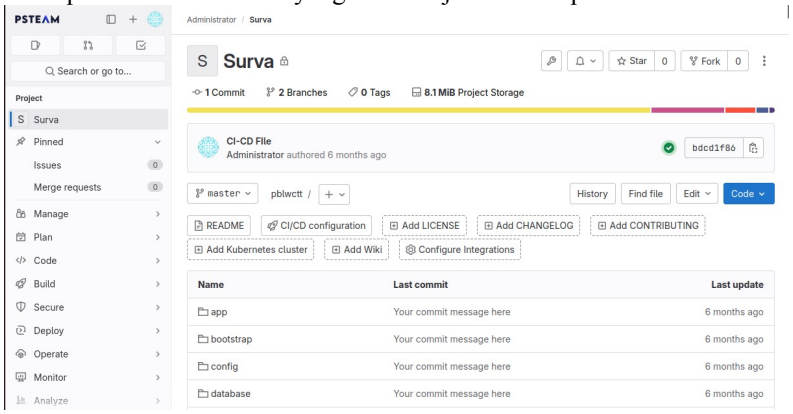
3.2. Pengoperasian

3.2.1. Metode Pengujian

Metode pengujian yang dilakukan adalah dengan menguji langsung *Pipeline* menggunakan terminal pada sistem operasi Ubuntu yang di dalam virtual box, menggunakan salah satu repositori pada Gitlab PSTeam yaitu Repositori pengembangan perangkat lunak Surva.

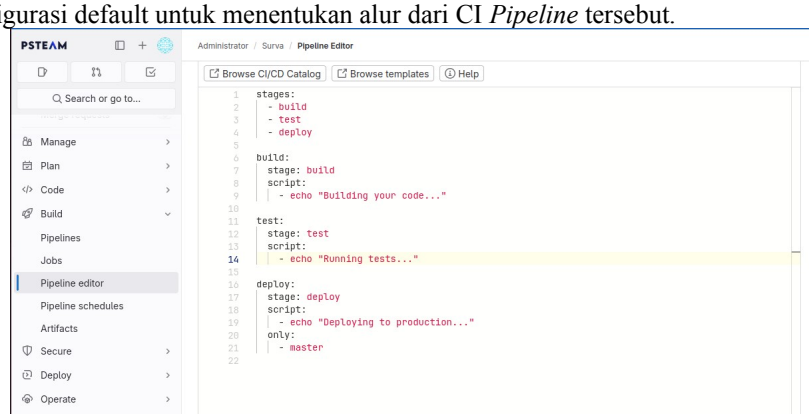
3.2.2. Pengujian konfigurasi Alur Pipeline

Pada tahap ini hasil implementasi sebelumnya dijalankan untuk melakukan pengujian pada hasil implementasi tersebut, repositori pada Gitlab PSTeam yang akan diuji adalah Repositori Surva.



Gambar 9. Repositori yang akan diuji

Pada repositori tersebut kan dibuat 1 file baru bernama gitlab-ci.yml dimana pengguna nantinya dapat melakukan pembuatan alur CI *Pipeline*, pengguna dapat membuat sendiri konfigurasi sesuai kebutuhan atau dapat mengikuti konfigurasi default untuk menentukan alur dari CI *Pipeline* tersebut.



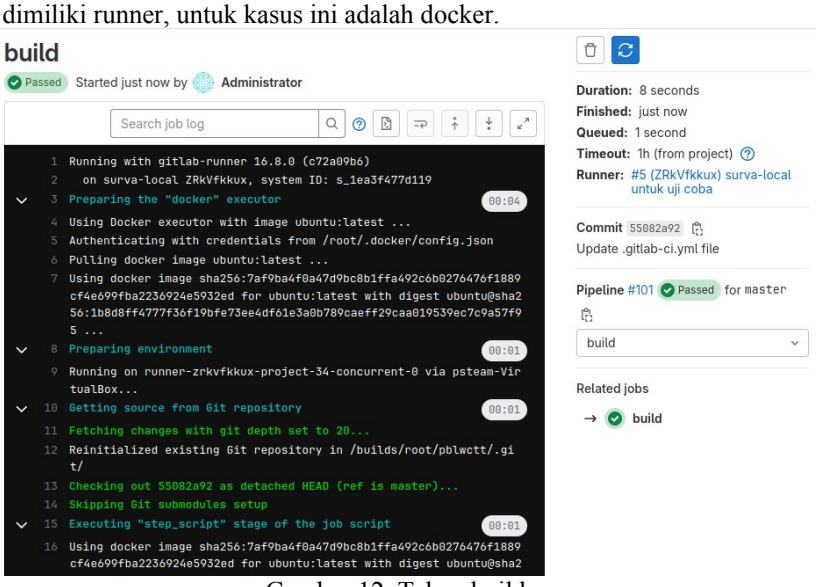
Gambar 10. Konfigurasi Alur Pipeline

Setelah Alur Pipeline di trigger maka proses CI/CD akan berjalan dengan menggunakan runner yang tersedia, yaitu runner yang sudah di registrasi sebelumnya, *Pipeline* ini terdiri dari 3 tahap, yaitu build, test dan deploy yang akan berjalan secara berurutan, selain itu status *Pipeline* juga akan memberikan pihak pengembang informasi apakah proses *Pipeline* masih berjalan, berhasil atau gagal.

Gambar 11. Status awal *Pipeline*

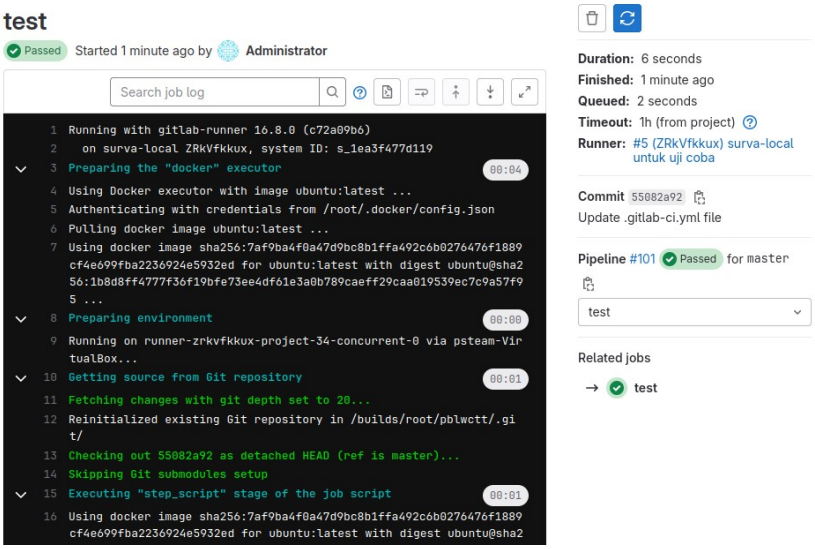
3.2.3. Pengujian tahap *Pipeline*

Tahap awal dari *Pipeline* tersebut adalah tahap build dimana *source code* pda repositori Surva akan di bangun kembali kedalam environment baru untuk dilakukan pengujian pada tahap selanjutnya menggunakan *executor* yang dimiliki runner, untuk kasus ini adalah docker.



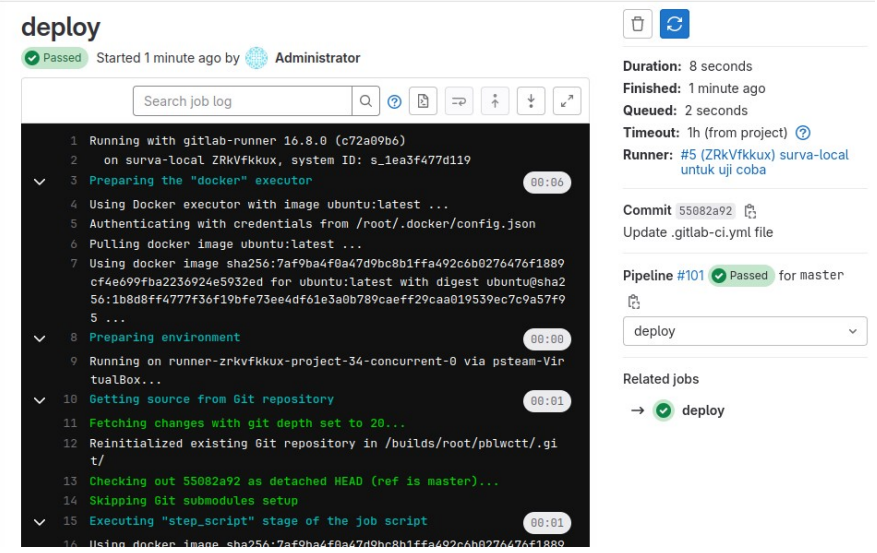
Gambar 12. Tahap build

Tahap selanjutnya pada *Pipeline* CI tersebut adalah tahap test, dimana *source code* yang sudah di build kembali dalam environment terisolasi akan di uji integritasnya serta adanya *syntax error* untuk nantinya di notifikasi ke pengembang untuk melakukan perbaikan, jika tahap test gagal maka *Pipeline* akan berhenti di tahap test dan dilakukan debugging pada *syntax error* yang diberikan, sedangkan ketika tahap test berhasil maka akan masuk ke tahapselanjutnya, yaitu tahap deploy.



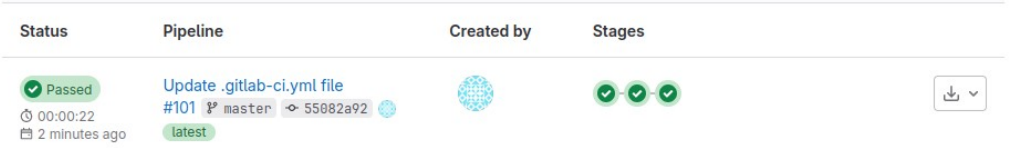
Gambar 13.Tahap test

terakhir pada Alur Pipeline tersebut adalah tahap Deployment dimana *source code* yang sudah diuji dan lolos akan di deploy kembali ke branch repositori sebelumnya, dan juga bisa dilakukan request merging kepada branch utama dari repositori tersebut.



Gambar 14. Tahap deploy

Setelah proses *Pipeline* CI/CD berhasil maka akan dinotifikasi kepada pengembang dan mendapatkan status passed dan ketiga tahapnya berhasil, pada tahap ini proses *Pipeline* CI/CD sudah selesai dan Runner yang menjalankannya akan tersedia kembali untuk digunakan pada *Pipeline* lainnya.



Gambar 15. Status akhir *Pipeline*

3.3. Optimasi

Tahap ini adalah tahap terakhir di mana proses CI/CD yang sudah berhasil dioperasikan dimonitor dievaluasi kembali, setelah dilakukan evaluasi ulang pada konfigurasi pada gitlab-ci.yml diputuskan untuk

membuat konfigurasi default yang nantinya menjadi dasar Alur Pipeline CI/CD yang akan digunakan, pengguna juga dapat mengganti konfigurasi tersebut sesuai kebutuhan project.

```yaml
stages:
  - build
  - test
  - deploy

before_script:
  - echo "Setting up environment..."

build:
  stage: build
  script:
    - echo "Building"


test:
  stage: test
  script:
    - echo "Callibrating The Code"

deploy:
  stage: deploy
  script:
    - echo "Deploying"
```

Gambar 16. konfigurasi default untuk gitlab-ci.yml

4. KESIMPULAN DAN SARAN

Polibatam Software Team akan terus berkembang seiring waktu, begitu pula dengan kebutuhan pengembangan perangkat lunak yang semakin kompleks ke depannya. Diharapkan dengan adanya CI/CD pada website repositori Gitlab PSTeam dapat membantu dan mempermudah PSTeam dalam mempersingkat waktu pengembangan perangkat lunak dengan mengurangi waktu pengembang dalam mencari error pada *source code* secara manual dan dapat memprioritaskan bagian lain yang lebih krusial dalam proyek pengembangan perangkat lunak. Selain itu dengan adanya kustomisasi dalam konfigurasi Runner dan Alur Pipeline juga dapat memberikan fleksibilitas dalam proses CI/CD sesuai kebutuhan proyek pengembangan perangkat lunak yang berbeda-beda, Berdasarkan hasil pengujian yang telah dilakukan, Implementasi *Pipeline* CI/CD Pada Gitlab PSTeam sudah dapat berjalan sesuai dengan yang diharapkan, runner dapat dibuat dan diregistrasi pada sistem operasi Ubuntu di dalam Virtual Box dan dapat dibagi menjadi 2 jenis, Runner dengan tag dan Runner tanpa tag, keduanya dapat menjalankan *Pipeline* yang membutuhkannya dengan pembeda Runner dengan tag hanya bisa menjalankan *Pipeline* dengan tag yang sama pada konfigurasi Alur Pipeline, sedangkan Runner tanpa tag bisa menjalankan semua *Pipeline* yang membutuhkannya, kendala terbesar yang dihadapi sejauh ini adalah menentukan konfigurasi Alur Pipeline yang bisa dijadikan standar dalam semua proyek repositori sebelum nantinya dapat di modifikasi sesuai kebutuhan oleh pengembang.

Untuk pengembangan ke depan, penulis dapat membuat standar konfigurasi Alur Pipeline yang lebih baik, menjadi dasar dari setiap proyek untuk kemudian dapat disesuaikan lebih lanjut oleh pengembang. Selain itu, perlu dilakukan uji perbandingan dan analisis terhadap proses pembuatan dan registrasi Runner pada sistem operasi lain, terutama Windows, selain Ubuntu. Dengan demikian, akan meningkatkan kesesuaian dan kinerja proyek di berbagai platform.

DAFTAR PUSTAKA

[1] Arefeen, M. M. (-, - -). Continuous Integration Using Gitlab. Diambil kembali dari Mendeley: https://www.mendeley.com/catalogue/77534141-996b-3168-bd2d-3e54234fe0e0/

[2] Ari Purno Wahyu, I. G. (2021, December 15). Implementasi Continuous Integration dan Continious Deployment Pada Aplikasi Learning Management System di PT. Millennia Solusi Informatika. Diambil kembali dari Jitter: https://journal.widyatama.ac.id/index.php/jitter/article/view/744

[3] Wijayanto, A. F. (2021, Oktober 28). Implementasi Continous Integration/Continous Delivery Menggunakan Process Manager 2 (Studi Kasus: SIAKAD Akademi Keperawatan Bina Insan). Diambil kembali dari Ejournal Ikado: https://ejournal.ikado.ac.id/index.php/teknika/article/view/400

[4] Dedi Dwi Saputra, D. D. (2024, February 7). Implementasi Private Cloud Storage dengan Menggunakan Owncloud dan Linux Ubuntu Pada Virtualbox Oracle. Diambil kembali dari Jurnal Komputer Antartika: https://ejournal.mediaantartika.id/index.php/jka/article/view/232

[5] Elfrin Erawan, M. S. (2023, August 16). Image based Ubuntu operating system using packer solutions. Diambil kembali dari Gema Wiralodra: https://www.gemawiralodra.unwir.ac.id/index.php/gemawiralodra/article/view/475

[6] Fernandes, P. A. (2023, November 15). Development of an ERP with CI/CD, Authentication and System Audit. Diambil kembali dari P.PORTO: https://recipp.ipp.pt/handle/10400.22/24284

[7] Leonel Hernandez, G. J. (2018, May 17). Design and Validation of a Scheme of Infrastructure of Servers, under the PPDIOO Methodology, in the University Institution - ITSA . Diambil kembali dari ResearchGate: https://link.springer.com/chapter/10.1007/978-3-319-91186-1_38

[8] Mohammed Shamsul Arefeen, M. S. (2019, September 11). Continuous Integration Using Gitlab. Diambil kembali dari URNCST Journal: https://www.urncst.com/index.php/urncst/article/view/152

[9] Muhammad Aditya, S. H. (2022, September 12). Perancangan Aplikasi Repository Skripsi Universitas Amir Hamzah Berbasis Web. Diambil kembali dari Remik: https://jurnal.polgan.ac.id/index.php/remik/article/view/11781

[10] NB Sellevåg, Y. S.-J. (2022, - -). Exploring possibilities for GitLab as a Learning Management System. Diambil kembali dari Google Scholar: https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/3003597/no.ntnu:inspera:106261571:112575889.pdf?sequence=1

[11] Putra.A, A. (2021, August 7). Implementasi High Availability Pada SERVER Layanan Web Dengan Cluster Computing. Diambil kembali dari Polibatam: https://if.polibatam.ac.id/tugas-akhir/viewer.php?file=dXBsb2Fkcy90dWdhc19ha2hpci80MzExNTExMDE5LzQ1MF9SYW5jYW5nXzIwMjEwOTI5LnBkZg==

[12] Rendy Wijaya, A. P. (2022, November 16). ImplementasiCI/CD Untuk BUIDDAN Deploy Website Dengan Docker Runner Pada Organisasi Belajar LINUX ID. Diambil kembali dari Ejournal Jak-Stik: https://ejournal.jak-stik.ac.id/index.php/sentik/article/view/3243