IMPLEMENTASI CI/CD UNTUK BUIDDAN DEPLOY WEBSITE DENGAN DOCKER RUNNER PADA ORGANISASI BELAJAR LINUX ID

Rendy Wijaya, Andri Prasetyo dan Dian Wahyuningsih
STMIK PPKIA Pradnya Paramita (STIMATA)
Jl. Laksda Adi Sucipto 249A, Malang, Jawa Timur
rendy.case@gmail.com, {andri, dian.wahyuningsih}@stimata.ac.id

ABSTRAK

Belajar Linux ID atau BLiD adalah komunitas yang terdiri dari anggota yang memiliki ketertarikan yang sama yaitu hal-hal yang berbau teknologi, seperti linux, infra, frontend, backend dan UI/UX namun ada keterlambatan dalam bidang teknologi pada proses melakukan buid, test dan deploy yang masih menggunakan manual. Proses buid, test dan deploy secara manual itu sehingga komunitas BLiD berniat untuk merenovasi system informasi lebih baik dengan menggunakan system automasi CI/CD. CI/CD merupakan system automasi yang membantu developer dan tester dalam melakukan rilis maupun update software dengan lebih cepat. Dari hasil analisis diperoleh bahwa terdapat beberapa bagian dalam proses buid, test dan deploy yang memerlukan proses automasi untuk memudahkan dan mempercepat programmer dalam melakukan proses buid, test dan deploy website

Kata Kunci : *CI/CD, Automasi, Website*

PENDAHULUAN

Teknologi sudah semakin maju dan berkembang. Tujuan penggunaan dan pemanfaatan teknologi ini untuk membantu menyelesaikan pekerjaan dengan cepat dan tepat.

Belajar Linux ID atau BLiD adalah komunitas yang terdiri dari anggota yang memiliki ketertarikan yang sama yaitu hal-hal yang berbau teknologi, seperti *linux*, *infra*, *frontend*, *backend* dan *UI/UX*. BLiD sendiri sudah berdiri sejak 2018 sampai sekarang.

BLiD memiliki beberapa produk salah satunya adalah website belajarlinux.id, website ini berisi tentang tutorial atau catatan bertema *cloud computing* dan dasar-dasar *linux*. Website belajarlinux.id sebelumnya dibuat menggunakan *wordpress*, tetapi karena *wordpress* cukup berat dan berlebihan jika hanya digunakan untuk *blogging* dan juga wordpress mengharuskan menggunakan database yang dimana cukup rumit pemeliharaannya maka BLiD memutuskan untuk memigrasi websitenya ke *static file generator Jekyll* pada tahun 2020.

Permasalahan yang timbul pada website belajarlinux.id adalah dalam proses *development* dan *deploy* pada website yang masih menggunakan cara manual, programmer yang membuat perubahan pada *code* web melakukan *push* ke *repository*. Lalu programmer menghubungi sysadmin untuk melakukan *build* dan *test code* terbaru. Kemudian saat *test* berhasil sysadmin akan *mendeploy code* ke *production server*. Setelah *code* berhasil di *deploy* ke *production server* barulah perubahan yang dibuat programmer akan tampak pada website. Dalam proses *development* pada website yang menjadi permasalahan adalah saat sysadmin tidak dalam keadaan *standby* atau tidak dalam keadaan siap mengerjakan proses *deploy*, programmer tidak akan dapat menerapkan hasil perubahan *code* ke website. Selain metode manual yang cukup rumit prosesnya, waktu yang dibutuhkan juga cukup lama yaitu kurang lebih 2 hari kerja dalam satu kali proses *build, test* dan *deploy*.

CI/CD adalah proses yang berkelanjutan dan terus menerus dalam *software development*, mulai dari awal hingga *software* tersebut mencapai *customer* dan mendapatkan *feedback*. *Dengan* menggunakan *CI/CD* dan *executor docker runner* akan mempesingkat proses *buil, test* dan *deploy. Docker runner* sendiri adalah *tool* yang di *install* di *server* yang nantinya akan menjalankan proses *build, test* dan *deploy* secara otomatis. Proses *CI/CD* dapat

dilihat dengan dengan website dan *cli* atau *Command Line Interface* langsung dari server.

Maka untuk mengatasi masalah yang timbul perlu adanya pengembangan sistem berbasis teknologi informasi untuk mempermudah proses *development* yang dapat dikembangkan melalui implementasi CI/CD. Oleh karena itu dari latar belakang tersebut, penelitian ini mengambil judul **"Implementasi CI/CD Untuk *Build* dan *Deploy* Website Dengan Docker Runner Pada Organisasi Belajar Linux ID".**

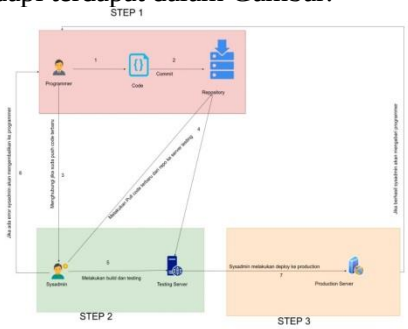
METODE PENELITIAN

Analisis Permasalahan

Pada bagian analisis ini menjelaskan tahapan untuk mengidentifikasi dan merumuskan pokok-pokok permasalahan secara terperinci. Adapun tahapan analisis masalah dapat diulas sebagai berikut:

Deskripsi Permasalahan

Dari hasil observasi didapatkan permasalahan yang dihadapi oleh komunitas Belajar Linux ID, alur dan masalah yang dihadapi terdapat dalam Gambar.



Gambar 1 . *Alur Kerja Sebelum Menggunakan CI/CD di Komunitas BliD*

Dari gambar diatas dapat diketahui bahwa setiap kali programmer melakukan perubahan pada *source code* maka akan melalui tiga langkah. Langkah pertama adalah melakukan *commit* dan *push* ke *repository*, Kemudian programmer harus menghubungi sysadmin untuk melakukan *build*, *test* dan *deploy*. Terdapat beberapa kemungkinan yang bisa terjadi.

Kemungkinan pertama adalah saat programmer dan syadmin ada di tempat dan sedang siap, maka yang terjadi adalah programmer meng *commit* kodenya ke *repository* kemudian programmer menghubungi sysadmin untuk melakukan *build*, *test* dan *deploy*. Setelah dihubungi programmer, sysadmin akan masuk ke *Testing server* kemudian sysadmin melakukan *pull code* dari *repository*. Selanjutnya sysadmin akan melakukan *build* dan *test* berdasarkan *code* yang di dapatkan, kemudian jika *code* gagal di *test* maka sysadmin menghubungi programmer jika kodenya gagal. Dan jika berhasil, hasil *code* yang sudah di *build* dan di *test* akan di teruskan ke *production server*. Permasalahannya adalah saat sysadmin melakukan *testing* dan terdapat *error* pada *code*, penyampaian *error code* tersebut dari sysadmin ke programmer akan sulit karena sysadmin hanya mendapat *error code* dari *terminal server*, dan tidak diketahui *error* berada di baris berapa.

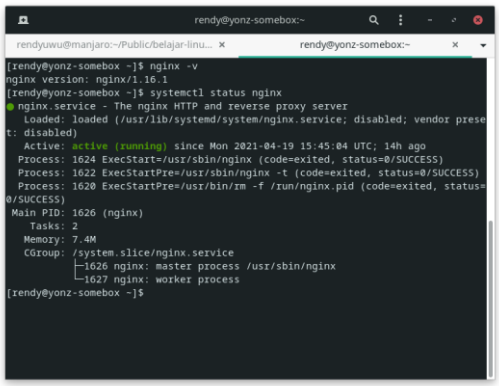
Kemungkinan kedua adalah saat sysadmin tidak di tempat atau tidak dalam keadaan siap, maka yang terjadi adalah programmer *commit* code ke *repository* kemudian programmer menghubungi sysadmin untuk melakukan *build*, *test* dan *deploy*, tetapi sysadmin sedang tidak ditempat atau tidak dalam ke adaan siap untuk melakukan *build*, *test* dan *deploy*. Akibatnya programmer harus menunggu terlebih dahulu sampai sysadmin siap dan akibatnya adalah code tidak langsung ter eksekusi ke production, dan programmer menjadi tidak tau apakah code yang dibuatnya sudah sesuai atau belum karena code masih belum di *testing* sysadmin.

Kemungkinan ketiga adalah saat programmer tidak ditempat, maka yang terjadi adalah setelah sysadmin melakukan *build* dan *test* code dan ternyata terdapat *error* maka sysadmin tidak dapat menyampaikan pesan *error* tersebut secara langsung kepada programmer dan akibatnya adalah *project* atau *source code* tersebut akan *stuck* dan tidak bias berlanjut ke tahap berikutnya sampai programmer bisa di hubungi. Hal ini tentu akan berdampak besar pada website utama, terlebih lagi jika website utama mengalami bug yang sangat penting dan harus di selesaikan dalam waktu dekat.

HASIL DAN PEMBAHASAN

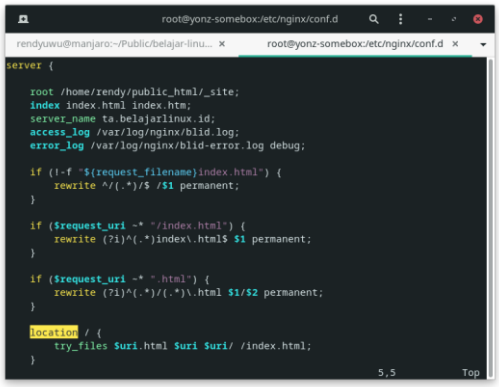
Pada tahap konfigurasi *CI/CD* memiliki beberapa tahapan diantaranya ialah installasi dan konfigurasi *nginx* (menampilkan hasil website ke internet), konfigurasi *docker* (container yang menjalankan *build, test & deploy*), installasi *gitlab-runner* (mengubungkan *server* ke gitlab agar dapat menjalankan *pipeline*), konfigurasi *runner (*mendaftarkan *server* ke *gitlab* sebagai *runner* yang akan menjalankan *job).*

Berikut merupakan hasil konfigurasi pada *load balancing NTH* yang disajian pada Gambar 2 sampai 6.



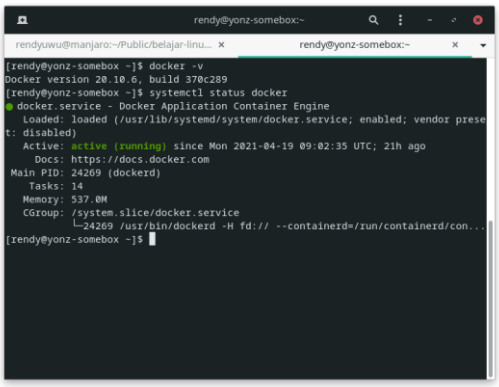
Gambar .2 .*Hasil installasi nginx di server*

Berdasarkan Gambar 1 dapat dijelaskan bahwa pada tahap ini dilakukan installasi *nginx* versi 1.16.1 pada *server*, dari Gambar 2 juga dapat diketahui jika *nginx* sudah berjalan secara normal dan sudah siap menerima *request* dari *browser user* dan siap mengambalikan sebuah *response* sesuai *request* dari *user*.



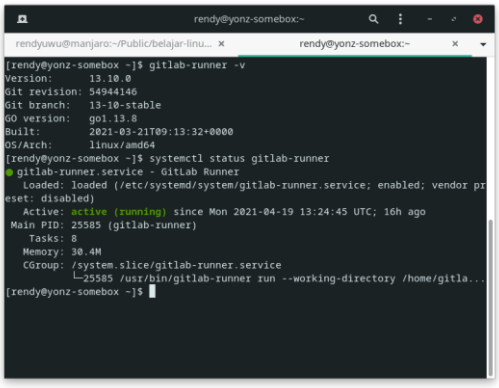
Gambar 3. *Hasil konfigurasi nginx di server*

Berdasarkan Gambar. 3 dapat dijelaskan bahwa pada tahap ini dilakukan konfigurasi *server block nginx* dengan keterangan bahwa alamat *host/server name* dari *domain* yang digunakan adalah ta.belajarlinux.id, dan pada konfigurasi di atas dilakukan *rewrite* pada *url* agar tidak menampilkan index.html pada akhir *url*, untuk port yang diganakan adalah 443 sehingga website sudah di berikan keamanan berupa *ssl*, dan bias diakses dengan protocol https.



Gambar. 4. *Hasil installasi Docker di server*

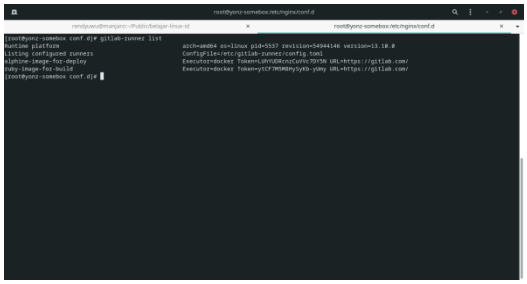
Berdasarkan Gambar .4 dapat dijelaskan bahwa pada tahap ini dilakukan installasi docker yang berguna untuk menjalankan container saat *server* melakukan *build, test* & *deploy* agar tidak merusak environment bawaan *server*.



Gambar. 5 *Hasil installasi gitlab-runner di server*

Berdasarkan Gambar.5 dapat dijelaskan bahwa pada tahap ini dilakukan installasi gitlab-runner versi 13.10.0 dengan keterangan bahwa gitlab-runner sudah

aktif(running) dan working directory berada pada /home/gitlab-runner.



Gambar 6 *Konfigurasi runner di server*

Berdasarkan Gambar.6 dapat dijelaskan bahwa pada tahap ini dilakukan konfigurasi runner dengan keterangan bahwa dalam konfigurasi ini di buat dua runner yang memiliki tugas masing-masing, runner pertama dengan tag ruby-image-for-*build* digunakan untuk melakukan *build* dan *testing* website, runner ini menggunakan image ruby untuk melakukan tugasnya. Runner kedua dengan tag alphine-image-for-*deploy* digunakan untuk *deploy* atau mengirim hasil *build* ke *server* untuk ditampilkan ke internet, runner ini menggunakan image alphine.

Pengujian

Pengujian yang dilakukan adalah pengujian pengukuran waktu *build, test & deploy* dari ketiga kondisi sebelum menggunakan sistem dan sesudah menggunakan sistem, yang nantinya akan dibandingkan dan disimpulkan apakah sistem yang di usulkan lebih efisien.

Pengujian Metode Manual

Pengujian metode manual adalah pengujian dengan menghitung total estimasi waktu dari kondisi satu, dua dan tiga, untuk waktu yang dihitung adalah waktu jam kerja yaitu 8 jam dalam sehari, jadi jika proses manual membutuhkan waktu dua hari maka waktu yang dihitung adalah 16 jam. Kemudian dari waktu tersebut akan di tarik kesimpulan rata-rata waktu yang dibutuhkan programmer jika menggunakan metode manual.

Terdapat tiga kondisi yang di uji antara lain, kondisi 1 adalah saat programmer dan sysadmin sedang berada di tempat dan sedang siap mengerjakan tugas masing-masing. Kemudian kondisi 2 adalah saat sysadmin tidak ditempat atau sedang tidak siap mengerjakan tugas nya, tetapi programmer dalam keadaan siap. Dan kondisi ketiga adalah saat programmer tidak ditempat atau sedang tidak siap tetapi sysadmin dalam keadaan siap. Estimasi waktu dari setiap kondisi disajikan pada Tabel 1 berikut.

Tabel. 1 *Estimasi Metode Manual*

Proses	Kondisi Satu/Menit	Kondisi Dua/Jam	Kondisi Tiga/Jam
BUILD & Test	15	8	8
Deploy	15	8	8
Total	30	16	16
Rata - Rata	39.000 Detik atau 10 Jam		

Pengujian Metode CI/CD

Pengujian Sistem CI/CD ini dilakukan dengan melakukan 10x proses CI/CD untuk menambil nilai rata-rata waktu yang dihabiskan untuk melakukan *build, test & deploy*. Karena setiap CI/CD berjalan waktu yang dibutuhkan tidak akan sama di karenakan ada toleransi waktu keterlambatan jika terjadi beberapa hal seperti, internet sedang lambat, *server* sedang sibuk dan *gitlab down*(kemungkinan kecil). Estimasi waktu dengan sistem CI/CD disajikan pada Tabel 2 berikut.

Tabel 2 *Estimasi Metode CI/CD*

Pengujian	*Build & Test/Detik*	*Deploy/Detik*	Toleransi/Detik
Pengujian 1	65	13	24
Pengujian 2	26	16	51
Pengujian 3	29	15	51
Pengujian 4	26	17	35
Pengujian 5	24	16	51
Pengujian 6	15	0	13
Pengujian 7	25	17	33
Pengujian 8	25	15	50
Pengujian 9	23	15	46
Pengujian 10	22	15	51
Total	302	139	405
Rata - Rata	282 Detik atau 4.7 Menit		

Perbandingan Antara Metode Manual Dan Sistem CI/CD

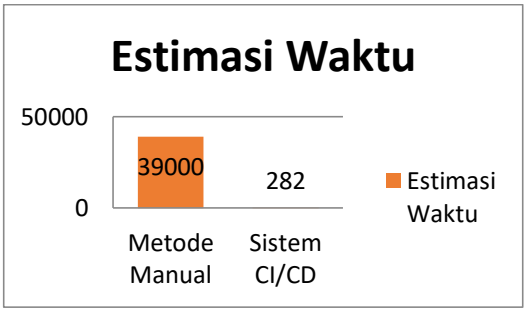
Perbandingan antara metode manual dan sistem CI/CD di simpulkan dari hasil pengujian yang telah dilakukan pada Tabel 1 dan Tabel 2. Hasil perbandingan ini akan disajikan pada Tabel 3 berikut.

Tabel 3 *Perbandingan Metode Manual dan Sistem CI/CD*

Metode	Estimasi Waktu
Manual	39.000 Detik
Sistem CI/CD	282 Detik
Efisiensi Waktu	138 kali lebih cepat

Berdasarkan Tabel 3 dapat disimpulkan jika rata-rata waktu yang dibutuhkan metode manual adalah 39.000 detik dan rata–rata waktu yang dibutuhkan metode *CI/CD* adalah 282 detik, Jadi jika rata-rata waktu metode manual dibagi rata-rata waktu metode CI/CD maka akan menghasilkan nilai 138 yang artinya metode CI/CD lebih cepat 138 kali.

Berikut hasil grafik dari perbandingan estimasi waktu antara metode manual dan sistem CI/CD disajikan pada Gambar 7.



Gambar 7 *Estimasi Waktu Metode Manual Dan Sistem CI/CD*

Berdasarkan hasil uji coba Estimasi Waktu seperti dalam Tabel 3 dan Gambar 6 terlihat bahwa metode CI/CD lebih menghemat waktu. Rata-rata waktu yang dibutuhkan metode manual adalah 10 Jam atau 39.000 Detik, sedangkan metode CI/CD hanya membutuhkan rata-rata waktu 282 Detik atau 4.7 Menit.

PENUTUP

Kesimpulan

Metode CI/CD dapat mengungguli metode manual dengan rasio yang sangat jauh, pada metode manual rata-rata waktu yang dibutuhkan untuk sekali proses *build, test & deploy* adalah 10 Jam atau sama dengan 39.000 Detik. Jika dibandingkan dengan metode dengan sistem CI/CD rata-rata waktu yang dibutuhkan untuk sekali proses *build, test & deploy* adalah 4.7 Menit atau sama dengan 282 Detik.

Berdasarkan hasil diatas maka dapat di simpulkan bahwa metode CI/CD telah memenuhi tujuan:

1) Memberikan solusi terkait dengan masalah-masalah pada proses development dan *deploy* di website belajarlinux.id.
2) Metode CI/CD lebih cepat 138 kali dari metode manual.

Saran

Hasil dari penelitian ini ialah dengan menerapkan metode CI/CD pada website belajarlinux.id sehingga dapat membuat automasi pada proses *build, test & deploy* sehingga dapat menigkatkan efisiensi waktu programmer. Namun metode ini memiliki kekurangan yaitu pada *production server* masih menggunakan *web server nginx* yang diinstall pada *server* secara langsung sehingga saat website mendapat request yang besar atau saat pengunjung website sangat banyak maka website akan lambat dan juga akan sulit untuk *scale up* website karena harus menambah *server* baru dan harus mengkonfigurasi *load balance*. Oleh karena itu saran untuk penelitian ini adalah dengan mengembangkan *production server* untuk menggunakan teknologi *container* seperti *docker* dan *kubernetes.*

DAFTAR PUSTAKA

[1] Arefeen, M. S., & Schiller, M. (2019). *Continuous Integration Using Gitlab.*

[2] Cadavid, H. F. (2018). *Continuous Delivery Pipeline for Teaching Agile and Developing Software Engineering Skils.*

[3] Hong, P. H. (2019). *Building a React Native application with a CI/CD pipeline*.

[4] Isnawaty, S. A., & Saputra , R. A. (2018). Desain dan Implementasi Virtualisasi Berbasis *Docker* Untuk Deployment Aplikasi Web.

[5] Kubrakov, K. (2017). *Deployment and Testing Automation in Web Application Implementing Devops Practices in Production.*

[6] Poornalinga, K. S., & P. R. (2016). *Continuous Integration, Deployment and Delivery Automation in AWS Cloud Infrastructure*.

[7] Shahin, M., Babar, M. A., & Zhu, L. (2015). *Continuous Integration, Delivery and Development: A Systematic Review on Approaches. Tools, Challenges and Practices*.

[8] Tohirin, Utam, S. F., Widianto, S. R., & Mauludyansah, W. A. (2020). *Implementasi DevOps pada Pengembangan Aplikasi e-Skrinning Covid-19*