BAB II. TINJAUAN PUSTAKA

Penerapan CI/CD yang memberikan banyak dampak positif sudah dilaksanakan oleh banyak orang. Tinjauan Pustaka ini akan membandingkan penelitian yang telah dilaksanakan sebelumnya. Perbandingan ini diharapkan memberikan acuan dan menjelaskan keunikan penelitian yang ditulis.

Penelitian pertama bertujuan untuk mengimplementasikan CI/CD menggunakan Jenkins, Docker Gitlab, dan Portainer pada aplikasi React dan Node JS. Parameter yang dicari dari penelitian ini adalah waktu yang dibutuhkan untuk proses manual dan dibandingkan dengan proses CI/CD. Aplikasi yang ada akan di-*containerized* yang kemudian akan dirilis pada *virtual machine* dengan *operation system* ubuntu. Proses yang dilalui pada pengujian secara manual dimulai dari *code commit, pull code*, *containerized* lalu *running* dari *container*. Sedangkan, pengujian pada CI dan CD dibagi menjadi kedua hal tersebut, yaitu CI untuk proses build dan testing, lalu CD untuk proses *deployment*. Prosesnya sama, yaitu *code commit*, lalu CI dan dilanjutkan pada CD. Waktu yang didapatkan pada pengujian tersebut membuktikan bahwa CI/CD lebih baik untuk performa dari segi waktu. Waktu yang dibutuhkan untuk proses manual adalah delapan menit 48 detik, sedangkan setelah implementasi CI/CD didapatkan waktu dua menit 56 detik. [6]

Penelitian kedua bertujuan untuk melakukan penelitian tentang pengembangan CI/CD untuk pengujian performa dari *DevOps* tersebut melalui aplikasi yang dibuat di pendidikan tinggi sebagai tugas akhir. Penelitian ini mengimplementasikan CI/CD menggunakan jenkins, docker, kubernetes dan argocd. Penelitian ini dilakukan karena proses pengembangan secara tradisional atau manual terjadi secara repetitif dalam *developing, testing dan deploying*. Dalam pengembangan aplikasi dengan tim juga ditemukan masalah mengenai tim operasional harus menunggu *developer* untuk menyelesaikan testing sebelum rilis aplikasi. Setelah proses *testing* selesai, konflik sering ditemukan karena lingkungan pengembangan yang berbeda, yang

memperlambat proses rilis. Masalah-masalah ini dapat menyebabkan penundaan dalam memberikan rilis aplikasi kepada klien. Implementasi CI/CD menjadi jawaban untuk permasalahan tersebut dikarenakan proses pengembangan menjadi otomatis sehingga mempercepat proses dari perilisan aplikasi. [7]

Penelitian ketiga bertujuan untuk mengimplementasikan CI/CD menggunakan AWS CodePipeline, AWS CodeDeploy dan Amazon EC2. Objek penelitiannya berupa *academic administrative services.* Penelitian ini mendiskusikan mengenai implementasi dari CI/CD dalam pengembangan *academic information system application* di Paramadina University yang masih menggunakan proses manual dalam pengembangannya. Implementasi dari CI/CD diharapkan dapat lebih efektif dan kualitas produk menjadi lebih baik. Penelitiannya sendiri menggunakan metode kualitatif dengan mengobservasi data yang terkumpul. Hasil dari penelitian ini membuktikan bahwa Implementasi CI/CD mempercepat proses *deployment* aplikasi, meminimalisir kemungkinan adanya *bug*, dan tim pengembang menjadi lebih mudah dalam melakukan pembaruan dan perubahan kapan saja ketika dibutuhkan. [8]

Penelitian keempat bertujuan untuk mengimplementasikan *DevOps* pada aplikasi skrining. Penelitian ini membahas penerapan *DevOps* menggunakan *tools* yang tidak terlalu banyak, yaitu *Gitlab* untuk mengimplementasikan CI/CD. Penelitian ini memberikan solusi untuk masalah dari pengembangan *agile scrum* yang memakan waktu banyak dan menyebabkan rilisan tidak sesuai dengan jadwal. Solusi untuk mengotomatisasi *build, test,* dan *deploy* bagi *project* yang dijalankan di lingkungan SDLC Melalui penelitian ini menghasilkan kesimpulan bahwa implementasi CI/CD memudahkan penggabungan kode, kelancaran *build*, dan pemeriksaan kelayakan kode setiap kali terjadi *trigger* dari pengembangan aplikasi. Penelitian ini mengamati penggunaan dari sumber daya pada proses CI/CD. [9]

Penelitian kelima bertujuan untuk melakukan penelitian tentang penerapan CI/CD dengan mengembangkan aplikasi *web* melalui Docker, Jenkins, Sonarqube dan Kubernetes. Penelitian ini mengimplementasikan

CI/CD menggunakan keempat *tools* tersebut pada sebuah Aplikasi myITS *Single Sign On*, yaitu aplikasi dari kampus ITS untuk memudahkan mahasiswanya. Proses pengembangan dari aplikasi tersebut masih manual, sehingga penelitian dilakukan untuk membuat proses pengembangan menjadi otomatis. *Tools* yang digunakan seperti Docker untuk mengemas aplikasi menjadi *image* hingga *deploy* pada Kubernetes. Jenkins yang digunakan untuk mengintegrasikan lingkungan *development* dan *production* serta tempat *pipeline* berada. Sonarqube digunakan untuk analisis *source code*, serta Kubernetes sebagai tempat aplikasi di-*deploy* dan dikelola. [10]

Penelitian keenam bertujuan untuk melakukan penelitian tentang penerapan CI/CD dengan mengembangkan aplikasi *web* melalui CodePipeline, GitHub, AWS Elastic Beanstalk, Slack dan EC2. Penelitian ini mengimplementasikan CI/CD menggunakan *tools* tersebut pada sebuah aplikasi web WeShop, yaitu aplikasi toko *online* yang berfungsi untuk jual beli barang. Aplikasi menggunakan bahasa pemrograman PHP, PHP Framework, HTML, Python dan Node JS. Github digunakan untuk *source repository* yang kemudian diintegrasikan dengan Slack untuk *monitoring*. CodePipeline digunakan untuk integrasi antara Github dengan Elastic Beanstalk dan EC2 untuk kemudian dilakukan perilisan secara otomatis. Parameter yang dihitung adalah waktu setiap bahasa pemrograman yang ada dari aplikasi web WeShop. Kelima web dengan bahasa pemrograman yang berbeda, tetapi dengan tampilan dan fungsi yang sama kemudian dibandingkan secara waktu. [11]

Penelitian ketujuh bertujuan untuk mengimplementasikan CI/CD menggunakan Jenkins, Slack dan Docker. Objek penelitiannya berupa aplikasi web. Penelitian ini membahas mengenai implementasi dari CI/CD dan pengaruhnya terhadap sumber daya, seperti kesehatan mesin, CPU dan RAM. Implementasi CI/CD dimulai dengan mengatur Jenkins sebagai server CI/CD, yang secara otomatis akan membangun, menguji, dan mendistribusikan aplikasi web tersebut setiap kali ada perubahan pada kode sumber. Jenkins akan diintegrasikan dengan Docker untuk memastikan setiap dilakukan dalam lingkungan yang konsisten, serta dengan Slack untuk memberikan notifikasi *real time* kepada tim pengembang mengenai status *build* dan *deploy*. Penelitian

ini juga melakukan pengawasan dan pencatatan kinerja mesin untuk membandingkan dua skenario: penggunaan Docker dan tidak menggunakan Docker. Dalam skenario pertama, aplikasi dan Jenkins berjalan di dalam kontainer Docker, sementara pada skenario kedua, mereka berjalan langsung pada *host machine* tanpa *containerized*. [12]

Penelitian kedelapan bertujuan untuk membandingkan *deployment* secara manual dan CI/CD secara paralel dan berurutan. Objek penelitiannya berupa aplikasi *mobile*. Penelitian ini membahas mengenai implementasi dari CI/CD dan pengaruhnya terhadap efisiensi waktu. Pengujian dilakukan pada VM dan layanan *cloud* yang digunakan adalah Azure. Sedangkan, layanan CI/CD yang digunakan adalah Azure DevOps. Penelitian mengamati proses terjadinya *deployment* berdasarkan tahapannya dan dilakukan pada lingkungan yang sama dan setara. Penelitian menunjukkan bahwa penggunaan CI/CD menghemat waktu, tetapi penggunaan CI/CD secara paralel menghemat lebih banyak waktu. [13]

Penelitian kesembilan bertujuan untuk membandingkan CI/CD yang menggunakan GitHub Actions dan Jenkins. Objek penelitiannya berupa aplikasi web. Penelitian ini membahas mengenai implementasi dari CI/CD dan pengaruhnya terhadap waktu lalu membandingkan kedua metode antara GitHub Actions dan Jenkins. Penelitian mengamati proses terjadinya *deployment* berdasarkan tahapannya dan dilakukan pada lingkungan yang sama dan setara. Penelitian menunjukkan bahwa penggunaan CI/CD menghemat waktu, tetapi penggunaan CI/CD dengan GitHub Actions menghemat lebih banyak waktu daripada Jenkins. [14]

Penelitian kesepuluh bertujuan untuk membandingkan CI/CD yang menggunakan Gitlab dan Jenkins. Objek penelitiannya berupa Aplikasi *monolithic* Basu Daily Farm Admin *Website*. Penelitian ini membahas mengenai implementasi dari CI/CD dan pengaruhnya terhadap waktu lalu membandingkan kedua metode antara Gitlab dan Jenkins. Penelitian mengamati proses terjadinya *deployment* berdasarkan tahapannya dan dilakukan pada lingkungan yang sama dan setara. Penelitian menunjukkan bahwa penggunaan CI/CD menghemat waktu, tetapi penggunaan CI/CD

dengan Jenkins menghemat lebih banyak waktu daripada Gitlab. [15]

Berdasarkan berbagai penelitian yang sudah dilakukan, maka akan dilakukan implementasi CI/CD. Hal ini dilakukan untuk pembuktian dari performa CI/CD yang dikatakan lebih baik daripada dilakukan secara manual atau tradisional. Berikutnya akan dilakukan analisis dampak CI/CD yang diperoleh berdasarkan waktu, biaya, sumber daya dan penggunaan paralel. Analisis yang dilakukan ada empat dikarenakan kebanyakan penelitian hanya menganalisis dari segi waktu. Sedangkan, CI/CD juga bisa berdampak pada segi sumber daya yang digunakan dan biaya dalam pengembangannya. Penggunaan GCP, terutama Cloud Build juga masih jarang disinggung, bahkan berdasarkan 10 jurnal yang ada belum ada yang membahas Cloud Build. Sehingga, penelitian yang akan dilakukan menggunakan layanan GCP, terutama Cloud Build serta menganalisis dampak dari segi waktu, biaya, sumber daya dan penggunaan paralel untuk pembuktian dari efisiensi CI/CD.

Tabel 2. 1 Tabel Perbandingan Penelitian

	Objek Penelitian	Tempat Perilisan	Sistem Operasi Perilisan	Bentuk Rilisan Aplikasi	*Tools*	Parameter yang Diukur
Nurhayati [6]	Aplikasi React dan Node JS	*Virtual Machine*	Ubuntu	*Container Image*	Jenkins, Docker Gitlab, dan Portainer	Waktu (Perbandingan Manual dan CI/CD)
Jaeni, dkk [7]	Aplikasi Project Amikom Yogyakarta	Kubernetes	Tidak Disebutkan	*Container Image*	Jenkins, Docker, Kubernetes, Sonarqube, dan ArgoCD	Waktu, Kualitas Kode
Rendy Indriyanto, dkk [8]	Aplikasi ASIK Universitas Paramadina	AWS EC2	Tidak Disebutkan	*Web App*	Git, AWS CodePipeline, AWS CodeDeploy	Kualitas Kode
Tohirin, dkk [9]	Aplikasi E-Skrining Covid-19	Gitlab	Linux CentOS 7	*Container Image*	Gitlab dan Gitlab CI	Sumber Daya (CPU dan RAM)

Restu Agung Parama, dkk [10]	Aplikasi myITS *Single Sign On*	Rancher Kubernetes	Ubuntu 20.04.2 LTS	*Container Image*	Docker, Jenkins, SonarQube dan Kubernetes	Waktu, Kualitas Kode
Alde Alanda, dkk [11]	Aplikasi Web WeShop	AWS Elastic Beanstalk dan EC2	Tidak Disebutkan	*Web App*	CodePipeline, GitHub, AWS Elastic Beanstalk, Slack dan EC2	Waktu
Rismanda Tyas Kusumadewi, dkk [12]	Aplikasi Web (Tidak Disebutkan Nama)	Docker Node	Tidak Disebutkan	*Web App*	Jenkins, Slack, Docker	Sumber Daya
Andreas Dimas Setyoko, dkk [13]	Aplikasi Mobile	Meta	Windows	*File* APK	Azure DevOps	Waktu
Zaidan	Aplikasi Web	*Local*	Windows	*Container*	Github Actions, Jenkins,	Waktu, Kualitas

Zulhakim, dkk [14]		*Environment* (Docker Compose)		*Image*	Docker	
Alif Babrizq Kuncara, dkk [15]	Aplikasi *monolithic* Basu Daily Farm Admin Website	*Virtual Machine*	Ubuntu 20.04 LTS Gen 2	*Container Image*	Jenkins, Gitlab, Azure VM	Waktu
Vincentius Agung Prabandaru	Aplikasi Web *Project* UAJY	Compute Engine	Ubuntu 20.04 LTS	*Container Image*	Cloud Source Repositories, Cloud Build, Docker, Artifact Registry, Secret Manager	Waktu, Biaya, Sumber Daya