Institut Riset dan Publikasi Indonesia (IRPI)
MALCOM: Indonesian Journal of Machine Learning and Computer Science
Journal Homepage: https://journal.irpi.or.id/index.php/malcom
Vol. 4 Iss. 3 July 2024, pp: 911-925
ISSN(P): 2797-2313 | ISSN(E): 2775-8575

Efficiency Comparison of Multi-Environment CI/CD Processes Through Parallel and Sequential Implementations

Perbandingan Efisiensi Proses CI/CD Multi-Lingkungan melalui Implementasi Paralel dan Berurutan

Andreas Dimas Setyoko[1], Amalia Zahra[2]

[1,2]Computer Science Department, Master of Computer Science, Binus Graduate Program,
Bina Nusantara University, Indonesia

E-Mail: andreas.dimassetyoko@binus.ac.id[1], amalia.zahra@binus.edu[2]

Received Mar 20th 2024; Revised Apr 30th 2024; Accepted May 24th 2024
Corresponding Author: Andreas Dimas Setyoko

Abstract

This research addresses the application development issues at PT. Astra International Tbk. by employing an automated Continuous Integration/Continuous Deployment (CI/CD) system. Currently, Astra faces compilation and distribution problems conducted manually, where the process is time-consuming and configuration errors often occur, especially due to various environments in each application. The proposed solution is the implementation of CI/CD to automate the compilation and distribution processes for each application environment. CI/CD is one of the DevOps practices used to organize software development more efficiently. By leveraging CI/CD, the development team can experience the benefits of faster application compilation and distribution processes. This study compares sequential CI/CD implementation with parallel CI/CD. The research findings indicate that sequential CI/CD can reduce the required time by 33% compared to manual processes, while parallel CI/CD can reduce the required time by 79% compared to manual processes.

Keyword: Automation, Continuous Deployment, Continuous Integration, Deployment, Mobile Applications

Abstrak

Penelitian ini mengatasi masalah pengembangan aplikasi di PT. Astra International Tbk. dengan menggunakan sistem otomatis Continuous Integration/Continuous Deployment (CI/CD). Astra saat ini menghadapi masalah kompilasi dan distribusi yang dilakukan secara manual dimana proses yang dilakukan memakan waktu yang lama dan seringkali terjadi kesalahan konfigurasi terlebih terdapat berbagai macam *environment* dalam tiap aplikasi. Solusi yang diusulkan adalah implementasi CI/CD untuk otomatisasi proses kompilasi dan distribusi untuk setiap *environment* aplikasi. CI/CD adalah salah satu praktik DevOps yang digunakan untuk pengembangan perangkat lunak menjadi lebih terorganisir. Dengan memanfaatkan CI/CD, tim pengembang dapat merasakan manfaat dari proses kompilasi dan distribusi aplikasi yang lebih cepat. Penelitian ini membandingkan implementasi CI/CD berurutan dengan CI/CD paralel. Hasil penelitian menunjukkan bahwa CI/CD berurutan dapat mengurangi waktu yang diperlukan sebesar 33% dari proses manual, sedangkan CI/CD paralel dapat mengurangi waktu yang diperlukan sebesar 79% dari proses manual.

Kata Kunci: Aplikasi Mobile, Continuous Deployment, Continuous Integration, Kompilasi, Otomatis

1. PENDAHULUAN

PT. Astra International Tbk. atau yang lebih dikenal dengan Astra merupakan perusahaan swasta yang berkantor pusat di Jakarta dan didirikan pada tahun 1957. Saat ini Astra memiliki tujuh lini bisnis: Otomotif; Jasa Keuangan; Alat Berat, Pertambangan & Energi; Agribisnis; Teknologi Informasi; Infrastruktur dan Logistik; Property [30]. Dalam lini bisnis Teknologi Informasi sudah banyak produk aplikasi-aplikasi yang dihasilkan. Sebagai contoh untuk aplikasi mobile sendiri terdapat sekitar 100 aplikasi yang terbagi dalam aplikasi *internal* perusahaan untuk dapat digunakan oleh karyawan perusahaan, serta aplikasi pelanggan untuk dapat digunakan oleh masyarakat umum [30].

Disaat proses pengembangan aplikasi di Astra didapatkan beberapa fakta mengenai proses hingga ke tahap rilis. Pertama tiap-tiap pengembang menulis kode di komputer mereka masing-masing, kemudian kode tersebut dikompilasi lalu diuji, apabila tidak terdapat masalah maka semua perubahan kode tersebut akan

disimpan (*commit*) ke Git (*push*). Setelah itu kode akan disatukan (*merge*) lalu dikompilasi kembali, kemudian *file* .apk hasil kompilasi tersebut akan dikirim ke tim penguji [2]. Apabila terdapat masalah terhadap *file* .apk tersebut maka tim penguji akan melaporkannya ke pengembang dan akan dilakukan perbaikan oleh pengembang, apabila tidak terdapat masalah maka aplikasi tersebut siap untuk dirilis [12]. Perangkat lunak yang digunakan selama dalam pengembangan aplikasi tersebut adalah Android Studio sebagai *Integrated Development Environment* (IDE), Git sebagai *Version Control System* (VCS), Azure Devops sebagai layanan manajemen *repository*, Microsoft Teams sebagai media komunikasi antar anggota tim, dan Meta serta Google Playstore sebagai media untuk mendistribusikan *file* apk ke anggota tim yang akan melakukan uji coba aplikasi [1]. Aplikasi META berfungsi sebagai sarana distribusi resmi aplikasi-aplikasi yang dikembangkan oleh Astra dan group Astra untuk diunduh dan digunakan oleh pengguna sebagai karyawan Astra.

Selama proses pengembangan aplikasi tersebut ditemukan beberapa masalah, yaitu pengembang harus melakukan kompilasi aplikasi melalui komputer mereka dan menunggu proses tersebut hingga berhasil kemudian membagikan *file* apk ke tim penguji [9]. Dalam proses kompilasi manual ini akan menggunakan *memory* cukup besar sekitar 4 hingga 6 *Gigabyte* sehingga pengembang tidak dapat melakukan hal lain ketika melakukan kompilasi aplikasi, apabila membutuhkan kompilasi lebih dari 1 aplikasi maka proses yang dilakukan sangat berat dan berakibat proses kompilasi paralel akan berjalan semakin lama [15]. Proses kompilasi manual ini juga seringkali mengalami kegagalan disebabkan konfigurasi yang digunakan tidak sesuai dan berbeda antar pengembang dalam satu tim [6]. Selanjutnya untuk membagikan *file* .apk pengembang harus mengunggah file tersebut ke Meta melalui *content management system* (CMS) Meta secara manual. Setelah *file* apk diunduh dan diuji, maka tim penguji akan memberikan *feedback* melalui Jira. Dalam proses distribusi manual ini seringkali mendapat masalah yaitu apk yang didistribusikan tidak tepat dengan yang seharusnya didistribusikan, sehingga dapat terlihat proses pengelolaan .apk tidak berjalan optimal terlebih lagi aplikasi yang harus diperhatikan sangat banyak yakni sekitar 84 aplikasi [17].

Berdasarkan paparan masalah di atas, maka solusi yang ditawarkan adalah dengan membangun sistem yang dapat melakukan proses pembangunan kode yang terintegrasi proses kompilasi dan distribusi secara otomatis dengan metode *Continuous Integration / Continuous Deployment* (CI/CD). CI/CD adalah salah satu praktik DevOps yang digunakan untuk pengembangan perangkat lunak menjadi lebih terorganisir. Penggunaan CI/CD pada pengembangan aplikasi memberikan banyak manfaat yang akan langsung terasa oleh tim pengembang [26]. Dalam penelitian yang berjudul Implementasi CI/CD Pipeline Pada Framework Androbase Menggunakan Jenkins, integrasi aplikasi androbase dengan CI/CD pipeline dapat mempermudah proses deployment aplikasi pada PT. Andromedia yang mana pada sistem sebelumnya masih menggunakan cara manual namun dengan menggunakan CI/CD proses deployment menjadi lebih cepat dan otomatis. Dari hasil penelitian dapat disimpulkan bahwa dengan menerapkan sistem CI/CD pipeline dengan menggunakan aplikasi jenkins dapat mempermudah proses deployment aplikasi di mana proses deployment yang awalnya menggunakan cara manual dengan CI/CD dapat dilakukan secara otomatis cara ini juga dapat menghemat waktu dan juga dapat meminimalisir kesalahan [22]. Dalam penelitian berjudul Rancang Bangun Sistem Continuous Integration Pipeline Menggunakan Jenkins di PT. Emporia Digital Raya disimpulkan dengan adanya sistem CI/CD terbukti dapat membantu para pengembang yang ada pada PT. Emporia Digital Raya maupun pengembang lain dalam mengembangkan suatu aplikasi agar lebih cepat dan meningkatkan kualitas kode program yang sedang dikerjakan [18]. Selain itu, ada banyak pilihan *tool a*tau alat yang dapat digunakan untuk mendukung berjalannya proses CI/CD ini secara otomatis dan aman [2]. Dalam penggunaan CI/CD *pipeline* ini kode akan diuji coba secara bersamaan agar proses pengembangan perangkat lunak dapat berjalan dengan seimbang. Uji coba dilakukan dengan CI *tool*. *Feedback* atau *error* yang terjadi juga dapat diketahui lebih cepat sehingga tim pengembang pun dapat langsung menindaklanjuti *feedback* tersebut secepat mungkin. Proses rilis dari suatu aplikasi dapat mungkin dapat dipercepat. Hal itu disebabkan kode-kode yang terus digabungkan dan diterapkan ke dalam produk, sehingga aplikasi selalu dalam kondisi siap untuk dirilis kapanpun [15].

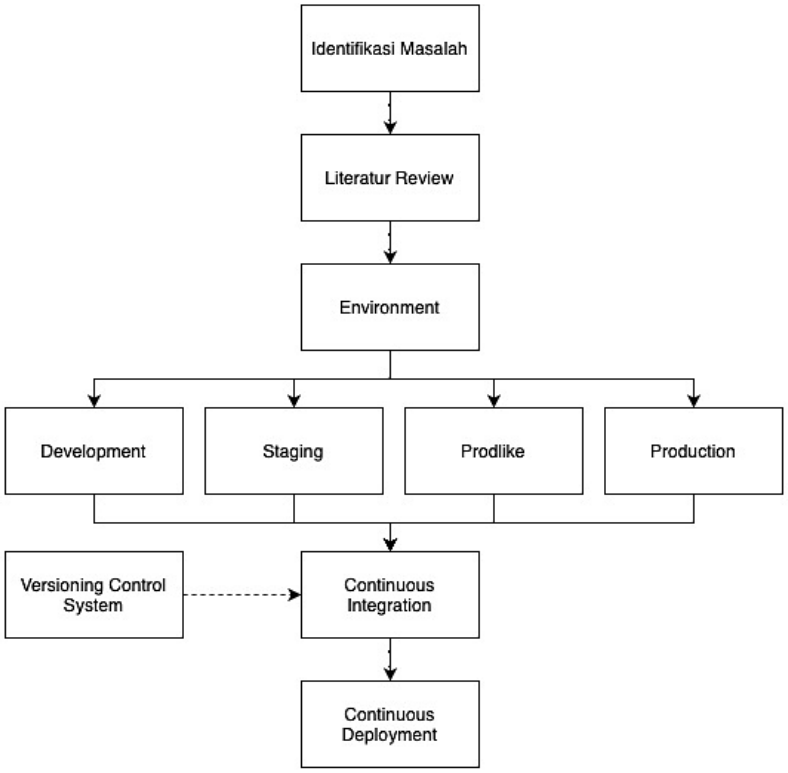
2. METODOLOGI PENELITIAN

2.1 Kerangka Pikir

Kerangka berpikir dalam penelitian CI/CD aplikasi mobile ini adalah suatu pandangan yang membantu peneliti untuk memahami bagaimana proses implementasi CI/CD dalam pengembangan aplikasi mobile. Kerangka ini memuat unsur-unsur yang saling terkait dan membantu peneliti untuk memahami alur logika yang digunakan dalam proses implementasi CI/CD. Kerangka pikir penelitian dapat dilihat pada Gambar 1.

Tahapan pertama dalam penelitian yang harus dilakukan adalah identifikasi masalah. Permasalahan yang diangkat pada penelitian ini adalah otomatisasi proses *build* dan *deploy* suatu aplikasi pada market menggunakan *pipeline* Azure DevOps yang di dalamnya terdapat juga beberapa pengujian untuk mengecek hasil *commit* yang dilakukan tim pengembang. Permasalahan yang dihadapi oleh PT Astra International Tbk diantaranya proses integrasi kode yang dilakukan secara manual oleh tim pengembang dapat memakan waktu dan menyebabkan konflik yang menghambat alur kerja. Proses *deployment* manual juga menyebabkan kesalahan dan membutuhkan waktu yang lama. Proses kerja sama antar tim pengembang dapat menyebabkan

konflik dan hambatan alur kerja. Proses distribusi yang dilakukan secara manual dapat menyebabkan variasi pada *set-up server* dan lingkungan. Proses rilis yang dilakukan secara manual dapat memakan waktu dan menyebabkan kesalahan.



Gambar 1. Kerangka Pikir

Dari permasalahan tersebut maka dalam penelitian ini peneliti melakukan studi literatur untuk memperkuat konsep praktik CI/CD, CI memastikan bahwa kode baru secara otomatis disatukan dengan kode lain dan dieksekusi pada setiap perubahan (*commit*) kode tiap pengembang. Disinilah peran dari *Versioning Control Sistem* (VCS) agar seluruh kode dari pengembang yang berbeda dapat disatukan dengan baik tanpa adanya konflik sekaligus tanpa adanya kode yang terlewat untuk digabungkan. CI juga memastikan bahwa setiap pengembang dapat bekerja pada kode yang sama dan memastikan bahwa perubahan mereka akan secara otomatis dikombinasikan dan dieksekusi. CD memastikan bahwa setiap *deployment* memiliki konfigurasi yang sama dan konsisten. CD juga memastikan bahwa rilis baru dapat dilakukan secara otomatis dan cepat.

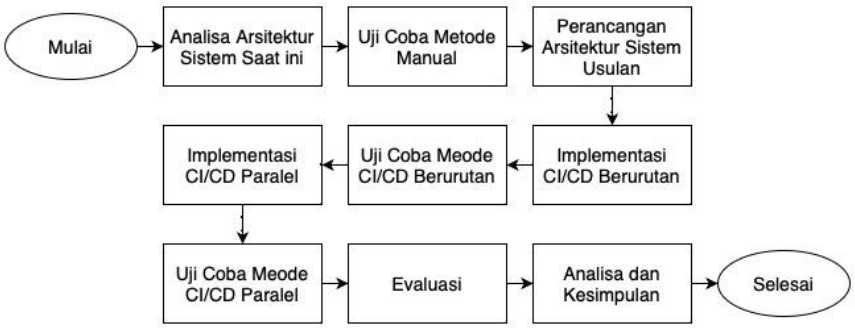
2.2 Tahapan Penelitian

Berdasarkan kerangka pikir penelitian pada Gambar 1, maka perlu dijelaskan lebih lanjut mengenai tahapan penelitian yang dapat dilihat pada Gambar 2.

Berdasarkan Gambar 2, setelah melakukan identifikasi masalah maka peneliti perlu mendapatkan solusi untuk menyelesaikan masalah tersebut. Oleh karena itu studi literatur dilakukan untuk mendapatkan data yang cukup untuk penelitian dan membantu peneliti untuk mendalami landasan teori yang berkaitan dengan permasalahan penelitian serta membantu peneliti untuk mengkaji kelebihan dan kekurangan hasil penelitian terdahulu. Studi literatur merupakan tahapan yang dibutuhkan oleh penulis agar dapat memperdalam pengetahuan tentang konsep-konsep dan cara mengimplementasikan ide yang akan dibuat. Pengetahuan tersebut diperoleh dari jurnal ilmiah dan sumber informasi yang berasal dari internet. Terdapat empat dasar teori yang menjadi acuan, yaitu *Software Development Life Cycle, Continuous Integration (CI), Continuous Deployment (CD),* dan *Version Control System (VCS).* Dalam studi literatur mendapatkan kesimpulan bahwa penerapan konsep CI/CD sangat berguna dan memudah tim pengembang dan operasional bekerja secara praktis. Proses *deployment* menggunakan CI/CD mampu mempersingkat proses dan meningkatkan kinerja. Selain itu CI/CD memberikan hasil yang lebih teliti dengan penemuan bug pada pengujian tersebut. Dalam studi literatur tersebut pula mendapat berbagai unsur dalam penerapan praktik CI/CD diantaranya: Peneliti harus memahami apa itu CI/CD dan bagaimana itu berfungsi dalam pengembangan aplikasi mobile; Peneliti harus memahami proses implementasi CI/CD dalam pengembangan aplikasi mobile, termasuk alur kerja dan tahapan-tahapan yang diperlukan; Peneliti harus memahami peran alat dan teknologi yang digunakan dalam

implementasi CI/CD, seperti Jenkins, Git, Docker, dan lain-lain; Peneliti harus memahami bagaimana manajemen konfigurasi dilakukan dalam CI/CD, termasuk penyimpanan konfigurasi, pemantauan, dan pembaruan; serta Peneliti harus memahami bagaimana automatisasi dilakukan dalam CI/CD, termasuk proses otomatisasi tes, integrasi, dan pemasangan.

Pada akhir tahapan penelitian terdapat evaluasi yang bertujuan untuk memastikan bahwa CI/CD dalam penelitian aplikasi mobile berjalan dengan efisien dan efektif, dan memastikan bahwa aplikasi mobile selalu up-to-date dan berkinerja baik. Langkah evaluasi CI/CD dengan menjalankan proses integrasi dan *deployment* secara berulang-ulang dan mengukur kinerja aplikasi mobile setiap kali proses berlangsung. Kemudian menganalisis data yang diperoleh dari proses evaluasi dan membandingkannya dengan tujuan dan kriteria yang telah ditentukan.



Gambar 2. Tahapan Penelitian

3. HASIL DAN PEMBAHASAN

3.1 Kompilasi dan Distribusi Secara Manual

Dalam proses kompilasi dan distribusi secara manual, peneliti melakukan perhitungan untuk mencatat tiap proses yang diperlukan dari awal proses hingga aplikasi selesai terdistribusi. Dalam kompilasi dan dsitribusi secara manual ini peneliti menggunakan resource berupa laptop kantor sedangkan untuk kompilasi dan distribusi secara CI/CD peneliti menggunakan resource server, perbandingan antar resource dapat dilihat pada tabel di bawah ini.

Tabel 1. Perbandingan resource antara proses manual dengan proses CI/CD

Spesifikasi	Resource Manual	Resource CI/CD
Edition	Windows 10 Pro	Windows Server 2016 Standard
Version	22H2	1607
OS Build	190.452.965	143.936.452
Processor	Intel® Core™ i5-5200U CPU @ 2.20GHz 2.19 GHz	Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz 2.59 GHz
Installed RAM	8.00 GB	8.00 GB
System type	64-bit operating system, ×64-based processor	64-bit operating system, ×64-based processor

Dalam penelitian ini dari sekian aplikasi mobile peneliti memilih satu aplikasi sebagai objek penelitian yakni aplikasi Assist H2 Reservasi. Assist H2 merupakan aplikasi PT. Astra International Tbk. dengan system operasi Android untuk memudahkan pembuatan Perintah Kerja Bengkel (PKB) yang sebelumnya dilakukan secara manual dimana seorang dengan peran Service Advisor (SA) mencatat secara manual pekerjaan servis, penggantian part, dan keluhan konsumen yang datang ke bengkel. Aplikasi ini hanya digunakan pada kendaran roda 2 dengan merek Honda. Proses bisnis yang dilakukan dalam aplikasi ini adalah pertama-tama konsumen datang ke bengkel untuk melakukan perbaikan kendaraan atau penggantian *sparepart* kendaraan, SA akan mencatat data kendaraan konsumen beserta data konsumen melalui aplikasi Assist H2. Dalam aplikasi ini pula SA akan memilih jenis servis yang diinginkan konsumen, SA juga dapat memilih *sparepart* melalui aplikasi. Setelah data berhasil disimpan maka mekanik akan melakukan proses perbaikan kendaraan, proses perbaikan akan termonitor dari aplikasi dan biaya perbaikan juga akan muncul dari aplikasi. Aplikasi ini juga dapat melihat history perbaikan kendaraan tersebut.

Proses-proses yang dilakukan pada proses kompilasi dan distribusi manual ada empat yakni melakukan *checkout* sumber kode, melakukan konfigurasi kode, melakukan kompilasi aplikasi, dan yang terakir adalah melakukan distribusi aplikasi. *Checkout* sumber kode adalah proses menyalin kode yang tersimpan pada repositori server ke laptop peneliti yang selanjutkan Salinan kode tersebut akan digunakan untuk kompilasi. Proses checkout kode secara manual dilakukan dengan memilih repository terlebih dahulu lalu menggunakan aplikasi terminal dengan menuliskan command git init serta git clone pada repository yang terpilih.

Selanjutnya dalam melakukan konfigurasi kode secara manual prosesnya adalah melakukan perubahan nilai parameter serta perubahan file yang sesuai pada *environment* yang dipilih. Apabila ingin melakukan kompilasi untuk aplikasi *environment* Development maka nilai parameter serta file harus menggunakan *environment* Development. Setiap *environment* memiliki nilai parameter berbeda-beda serta file yang berbeda-beda pula. Nilai parameter yang diubah antara lain parameter alamat server dan parameter debug untuk menghidupkan debug tools. Untuk file yang perlu diubah antara lain file keystore untuk melakukan signing aplikasi agar aplikasi tersebut tidak dapat di-debug, dan file Google Service untuk kebutuhan login serta menghidupkan fitur push-notification. Proses konfigurasi kode ini seluruhnya dilakukan pada aplikasi Android Studio.

Setelah sumber kode dilakukan konfigurasi untuk *environment* yang dipilih maka akan dilakukan kompilasi. Pengembang cukup menunggu kompilasi ini selesai karena proses kompilasi ini sepenuhnya dilakukan oleh aplikasi Android Studio. Proses kompilasi ini akan membutuhkan memory yang cukup besar sehingga laptop akan menyisakan memory yang sedikit untuk laptop pengembang dan hal ini akan menghambat pengembang untuk melakukan pekerjaan lain ketika menunggu proses kompilasi selesai.

Ketika proses kompilasi selesai maka akan menghasilkan sebuah file dengan format ekstensi .apk. File ini merupakan file instalasi aplikasi. Selanjutnya file ini akan didistribusikan pengembang melalui platform distribusi Meta. Hal yang perlu dilakukan adalah login pada web Meta, memilih aplikasi yang ingin dilakukan pembaharuan, melakukan unggah file .apk tersebut, dan setelah berhasil terunggah perlu dilakukan simpan pembaharuan. Setelah berhasil tersimpan maka aplikasi tersebut berhasil didistribusikan. Proses awal kompilasi dan distribusi secara manual ini digambarkan pada Gambar 3.

Gambar 3. Proses checkout sumber kode dalam CI/CD

Gambar 3. merupakan diagram proses kompilasi dan distribusi secara manual dimana pertama-tama dilakukan *checkout* sumber kode, kemudian selanjutnya melakukan konfigurasi kode, melakukan kompilasi dan yang terakhir adalah proses distribusi dengan mengunggah file ke Meta. Peneliti melakukan percobaan kompilasi dan distribusi secara manual sebanyak 5 kali untuk tiap-tiap *environment* Development, Staging, Prodlike, dan Production dengan hasil perhitungan waktu tiap-tiap prosesnya tercatat pada Tabel 2.

Tabel 2. Proses kompilasi dan distribusi aplikasi secara manual pada environment development.

Proses yang dilakukan	Waktu yang dibutuhkan		
	Minimum	Maksimum	Rata-Rata
Checkout kode	2 menit 45 detik	6 menit 32 detik	3 menit 30 detik
Konfigurasi kode	8 menit 12 detik	14 menit 5 detik	10 menit 11 detik
Kompilasi	64 menit 22 detik	80 menit 45 detik	70 menit 32 detik
Distribusi aplikasi	9 menit 14 detik	13 menit 1 detik	10 menit 47 detik

Tabel 2. merupakan hasil dari waktu yang dibutuhkan untuk tiap-tiap proses yang dilakukan untuk kompilasi dan distribusi aplikasi pada *environment* development. Proses ini memerlukan waktu rata-rata keseluruhan sekitar 95 menit 0 detik (1 jam 35 menit 0 detik) untuk menyelesaikan kompilasi dan distribusi secara manual. Waktu yang diperlukan bervariasi tergantung pada masing-masing langkah. Checkout kode adalah langkah paling cepat dengan waktu minimum 2 menit 45 detik dan maksimum 6 menit 32 detik. Sedangkan kompilasi adalah langkah yang paling lama dengan waktu minimum 64 menit 22 detik dan maksimum 80 menit 45 detik. Kompilasi kode adalah langkah yang paling memakan waktu, dengan rata-rata sekitar 70 menit 32 detik. Langkah ini adalah langkah di mana proses CI/CD sangat diperlukan untuk memangkas waktu.

Hasil proses kompilasi dan distribusi secara manual untuk *environment* Staging digambarkan pada Tabel 3, sedangkan untuk *environment* Prodlike pada Tabel 4, serta *environment* Production pada Tabel 5.

Tabel 3. Proses Kompilasi dan Distribusi Aplikasi Secara Manual Pada Environment Staging

Proses yang dilakukan	Waktu yang dibutuhkan		
	Minimum	Maksimum	Rata-Rata
Checkout kode	2 menit 38 detik	5 menit 11 detik	3 menit 45 detik
Konfigurasi kode	8 menit 48 detik	10 menit 43 detik	9 menit 12 detik

Proses yang dilakukan	Waktu yang dibutuhkan		
	Minimum	Maksimum	Rata-Rata
Kompilasi	68 menit 47 detik	76 menit 56 detik	70 menit 34 detik
Distribusi aplikasi	10 menit 8 detik	13 menit 27 detik	12 menit 19 detik

Tabel 4. Proses kompilasi dan distribusi aplikasi secara manual pada environment prodlike.

Proses yang dilakukan	Waktu yang dibutuhkan		
	Minimum	Maksimum	Rata-Rata
Checkout kode	3 menit 1 detik	5 menit 9 detik	3 menit 31 detik
Konfigurasi kode	8 menit 28 detik	11 menit 44 detik	10 menit 55 detik
Kompilasi	87 menit 35 detik	97 menit 44 detik	91 menit 3 detik
Distribusi aplikasi	11 menit 2 detik	12 menit 28 detik	11 menit 54 detik

Tabel 5. Proses kompilasi dan distribusi aplikasi secara manual pada environment production.

Proses yang dilakukan	Waktu yang dibutuhkan		
	Minimum	Maksimum	Rata-Rata
Checkout kode	3 menit 2 detik	4 menit 12 detik	3 menit 35 detik
Konfigurasi kode	9 menit 21 detik	11 menit 32 detik	10 menit 2 detik
Kompilasi	66 menit 42 detik	77 menit 56 detik	68 menit 51 detik
Distribusi aplikasi	10 menit 22 detik	13 menit 47 detik	11 enit 12 detik

3.2 Kompilasi dan Distribusi Secara Berurutan Menggunakan CI/CD

Pada sub-bab ini akan dibahas proses kompilasi dan distribusi aplikasi tiap-tiap *environment* dengan menggunakan CI/CD. Hal yang paling membedakan proses CI/CD ini dengan proses manual adalah keseluruhan proses CI/CD ini dilakukan di server dan tidak dilakukan pada laptop pengembang seperti yang dilakukan dalam proses manual. Kompilasi dan distribusi secara CI/CD ini juga terdapat 4 proses diantaranya proses *checkout* kode, melakukan konfigurasi kode, melakukan kompilasi aplikasi, dan yang terakir adalah melakukan distribusi aplikasi. Semua proses ini dilakukan secara otomatis dalam server CI/CD dengan membuat perintah-perintah dalam bentuk script yaml.

Proses checkout kode CI/CD merupakan proses membaca sumber kode yang dipilih yang nantinya akan diproses selanjutnya. Proses ini dilakukan pada server yang sama sehingga dapat memangkas proses manual dimana dalam proses manual diperlukan untuk mengunduh sumber kode ke laptop pengambang untuk diproses selanjutnya. Proses checkout kode ini cukup menggunakan perintah “checkout” seperti pada penggalan kode di bawah ini.

Source Code 1
```
jobs:
  - job: 'BuildAndPush'
    pool:
      name: 'Build-Android'
    displayName: 'Compile APK Development'
    steps:
    - checkout: ${{ parameters.appRepository }}
```

Perintah proses checkout pada penggalan kode di atas cukup menuliskan parameters appRepository dimana nilai dari parameter ini merupakan repository aplikasi yang ingin dikompilasi. Langkah selanjutnya adalah melakukan konfigurasi kode dimana prosesnya adalah melakukan perubahan nilai parameter serta perubahan file yang sesuai pada *environment* yang dipilih. Apabila ingin melakukan kompilasi untuk aplikasi *environment* Development maka nilai parameter serta file harus menggunakan *environment* Development. Setiap *environment* memiliki nilai parameter berbeda-beda serta file yang berbeda-beda pula. Proses konfigurasi kode dapat dilihat pada penggalan kode di bawah ini.

Source Code 2
```
- task: PowerShell@2
      displayName: Disable OutputFIleName
      inputs:
        targetType: 'inline'
        script: |
```

```
          ((Get-Content -Path $(build.sourcesdirectory)/PssH2-QMS-app/build.gradle) -
replace "outputFileName","//outputFileName") | Set-Content -Path
$(build.sourcesdirectory)/PssH2-QMS-app/build.gradle
          Write-Verbose "Changing completed..." -Verbose
```

Penggalan kode di atas merupakan salah satu konfigurasi kode untuk mengubah nilai dari parameter outputFileName. Proses tersebut dijalankan menggunakan library PowerShell yang akan menjalankan script di atas dimana script tersebut merupakan perintah untuk mengubah nilai outputFileName dengan nilai yang diharapkan secara otomatis. Setelah menjalankan seluruh konfigurasi kode, maka langkah selanjutnya adalah melakukan kompilasi. Kompilasi dalam CI/CD menggunakan perintah library Gradle, perintah ini dapat dilihat pada penggalan kode di bawah ini.

Source Code 3

```
- task: Gradle@2
      inputs:
        gradleWrapperFile: 'gradlew'
        tasks: 'assembleDevelopment'
        publishJUnitResults: false
        testResultsFiles: '**/TEST-*.xml'
        javaHomeOption: 'JDKVersion'
        sonarQubeRunAnalysis: false
```

Penggalan kode di atas untuk melakukan kompilasi secara CI/CD. Dalam input tersebut terdapat sebuah parameter task yang bersi assembleDevelopment. Task ini merupakan perintah untuk melakukan kompilasi sesuai dengan konfigurasi *environment* Development, apabila ingin melakukan kompilasi aplikasi *environment* lainnya maka parameter ini perlu disesuaikan sedangkan paramere input lainnya merupakan parameter bawaan library. Kompilasi ini dilakukan di server sehingga memory laptop pengembang tidak lagi diperlukan untuk kompilasi dan pengembang dapat melakukan pekerjaan lainnya secara maksimal.

Setelah selesai melakukan kompilasi maka akan dilakukan distribusi aplikasi. Hasil kompilasi itu akan menghasilkan sebuah file dengan ekstensi .apk. File ini mula-mula perlu disimpan dalam *storage server* dengan perintah copy file serta perintah publish artifact pada penggalan kode di bawah.

Source Code 4

```
- task: CopyFiles@2
      inputs:
        SourceFolder: 'PssH2-QMS-app/build/outputs/apk/development/debug'
        Contents: '**/*.apk'
        TargetFolder: '$(Build.ArtifactStagingDirectory)'

    - task: PublishBuildArtifacts@1
      inputs:
        PathtoPublish: '$(Build.ArtifactStagingDirectory)'
        ArtifactName: 'ArtifactDevelopment'
        publishLocation: 'FilePath'
        TargetPath: '\\A000S-ITCICD4\Drops\$(Build.DefinitionName)\$(Build.BuildNumber)'
```

Mula-mula terdapat 2 perintah yakni CopyFile untuk mencari file .apk hasil dari kompilasi dan selanjutnya perintah PublishBuildArtifact untuk menyalinnya dalam folder ITCICD04. File ini nantinya akan digunakan untuk diunggah ke Meta. Selanjutnya dalam proses unggah ke Meta maka perlu mengubah proses manual menjadi CI/CD menggunakan script terminal. Proses yang perlu diubah dalam mengunggah file ke Meta terdapat 3 proses yakni login ke dalam Meta, mengunggah file, dan menyimpan aplikasi yang digambarkan pada penggalan kode dibawah.

Source Code 5

```
- task: PowerShell@2
      displayName: Login Meta
      inputs:
        targetType: 'inline'
        script:
- task: PowerShell@2
      displayName: Upload File
      inputs:
        targetType: 'inline'
        script:
- task: PowerShell@2
```

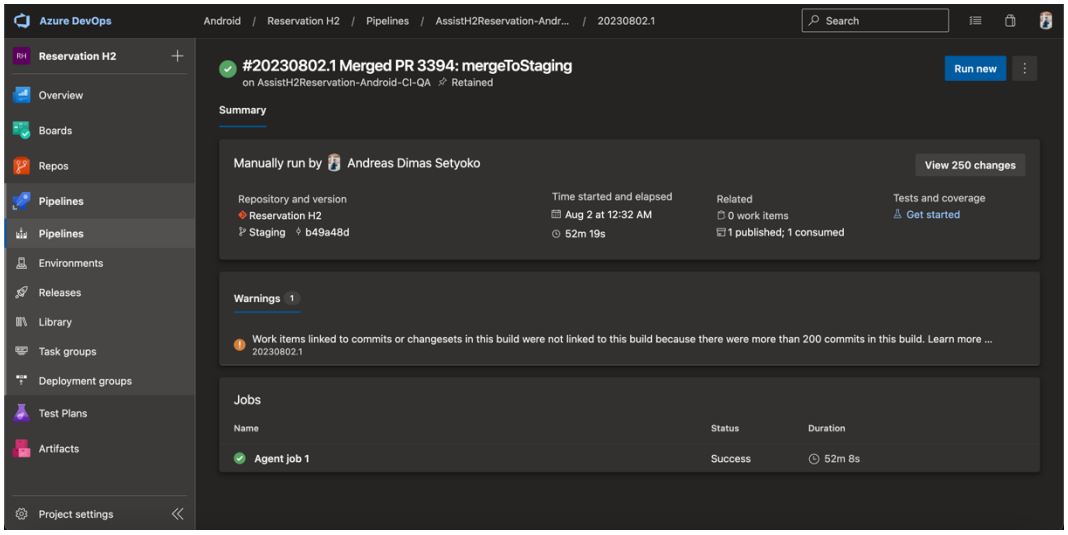
```
Source Code 5
      displayName: Update Apps
      inputs:
        targetType: 'inline'
        script:
```

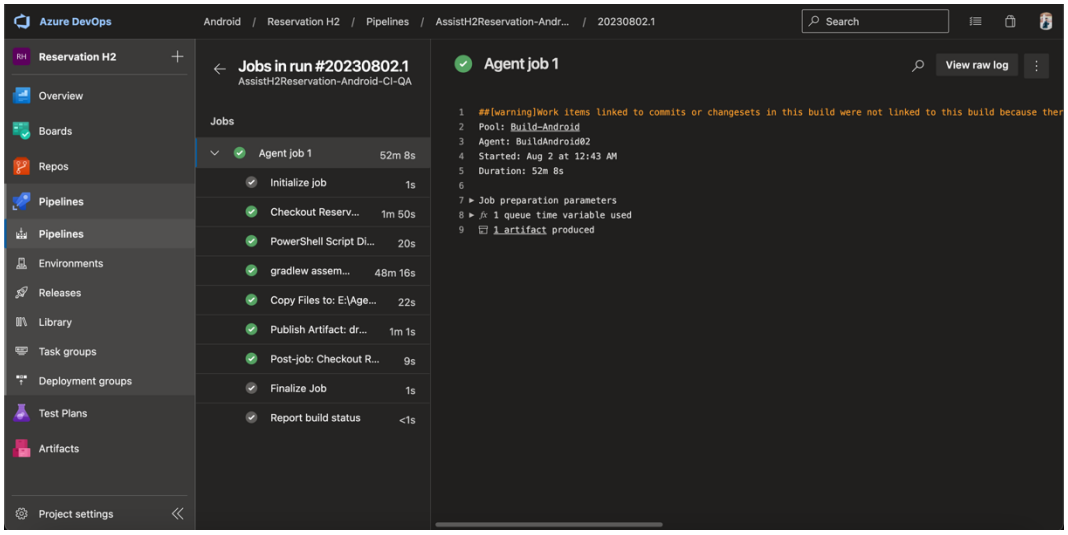
Penggalan kode di atas untuk melakukan distribusi dengan cara mengunggah file aplikasi ke Meta. Ketiga proses ini menggunakan library PowerShell yang akan menjalan perintah berdasarkan script yang ditulis. Peneliti melakukan percobaan kompilasi dan distribusi dengan CI/CD sebanyak 5 kali untuk tiap-tiap *environment* Development, Staging, Prodlike, dan Production dengan hasil perhitungan waktu tiap-tiap prosesnya tercatat pada Tabel 6.

Tabel 6. Proses Kompilasi dan Distribusi Aplikasi dengan CI/CD pada Environment Development.

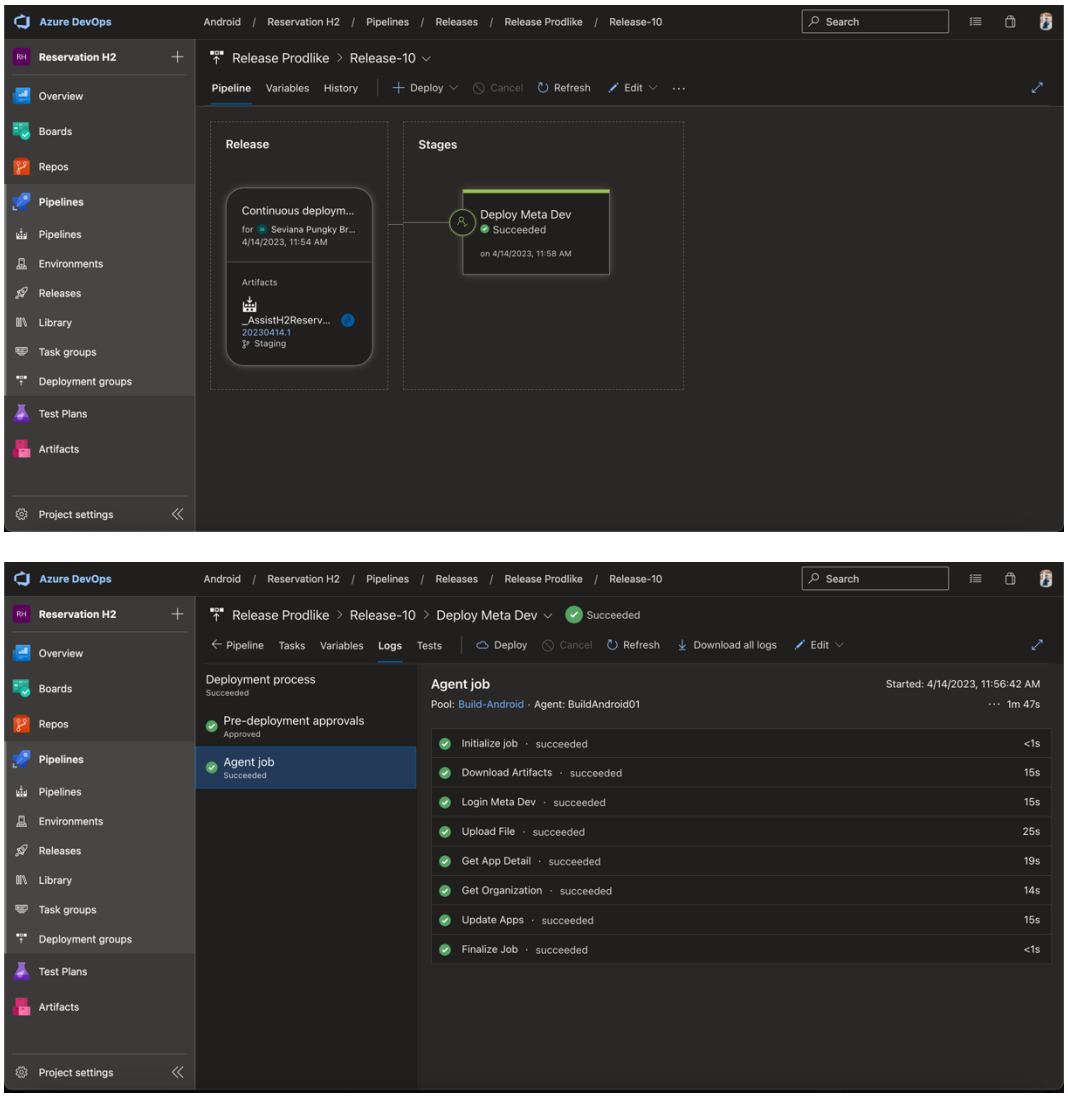
Proses yang dilakukan	Waktu yang dibutuhkan		
	Minimum	Maksimum	Rata-Rata
Checkout kode	1 menit 10 detik	1 menit 58 detik	1 menit 41 detik
Konfigurasi kode	30 detik	1 menit 12 detik	41 detik
Kompilasi	57 menit 12 detik	66 menit 47 detik	60 menit 41 detik
Distribusi aplikasi	2 menit 1 detik	3 menit 33 detik	2 menit 29 detik

Tabel 6. menunjukkan hasil dari penerapan CI/CD dalam proses kompilasi dan distribusi aplikasi pada *environment* Development. Proses checkout kode merupakan tahap di mana kode aplikasi diunduh dari repositori sumber (source repository) ke laptop pengembang. Hasilnya menunjukkan bahwa waktu minimum yang dibutuhkan untuk tahap ini adalah 1 menit 10 detik, waktu maksimum adalah 1 menit 58 detik, dan rata-rata waktu yang dibutuhkan adalah 1 menit 41 detik. Proses konfigurasi kode merupakan tahap dimana dilakukan penyesuaian konfigurasi aplikasi sesuai dengan *environment* yang dipilih. Waktu minimum yang diperlukan adalah 30 detik, waktu maksimum adalah 1 menit 12 detik, dan rata-rata waktu yang dibutuhkan adalah 41 detik. Proses kompilasi merupakan proses kompilasi kode sumber menjadi aplikasi yang dapat dijalankan. Waktu minimum yang dibutuhkan untuk tahap ini adalah 57 menit 12 detik, waktu maksimum adalah 66 menit 47 detik, dan rata-rata waktu yang dibutuhkan adalah 60 menit 41 detik. Kemudian proses terakhir merupakan distribusi aplikasi yang telah dikompilasi ke lingkungan pengembangan. Waktu minimum yang dibutuhkan adalah 2 menit 1 detik, waktu maksimum adalah 3 menit 33 detik, dan rata-rata waktu yang dibutuhkan adalah 2 menit 29 detik. Tampilan *user interface* proses CI/CD berurutan ditampilkan pada gambar 4 untuk CI dan gambar 5. untuk CD.





Gambar 4. User Interface CI Secara Berurutan



Gambar 5. User Interface CD Secara Berurutan

Pada gambar 4. dapat terlihat proses CI dilakukan dalam waktu 52 menit 8 detik dan ditampilkan pula tiap detil prosesnya beserta waktunya. Sedangkan pada gambar 5. terlihat proses CD dilakukan dalam waktu 1 menit 47 detik.

Hasil proses kompilasi dan distribusi dengan CI/CD untuk *environment* Staging digambarkan pada Tabel 7, sedangkan untuk *environment* Prodlike pada Tabel 8, serta *environment* Production pada Tabel 9.

Tabel 7. Proses kompilasi dan distribusi aplikasi dengan CI/CD pada environment staging.

Proses yang dilakukan	Waktu yang dibutuhkan		
	Minimum	Maksimum	Rata-Rata
Checkout kode	1 menit 2 detik	2 menit 12 detik	1 menit 47 detik
Konfigurasi kode	44 detik	1 menit 4 detik	49 detik
Kompilasi	48 menit 11 detik	56 menit 17 detik	53 menit 15 detik
Distribusi aplikasi	2 menit 26 detik	5 menit 39 detik	3 menit 45 detik

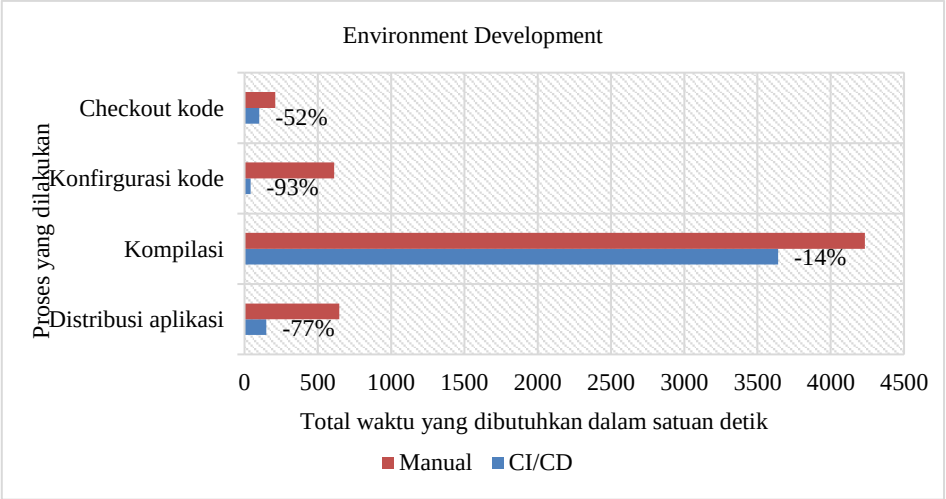
Tabel 8. Proses Kompilasi dan Distribusi Aplikasi dengan CI/CD pada Environment Prodlike.

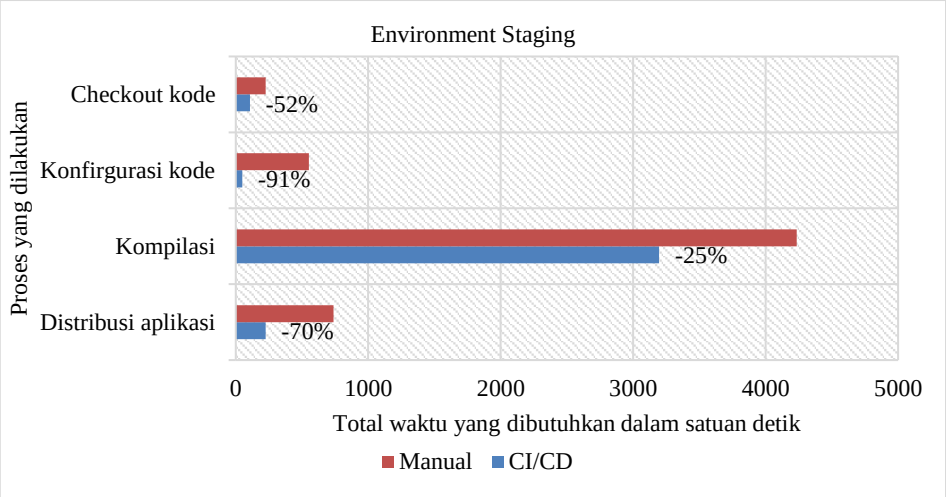
Proses yang dilakukan	Waktu yang dibutuhkan		
	Minimum	Maksimum	Rata-Rata
Checkout kode	2 menit 12 detik	3 menit 1 detik	1 menit 41 detik
Konfigurasi kode	1 menit 40 detik	3 menit 14 detik	2 menit 13 detik
Kompilasi	64 menit 14 detik	79 menit 33 detik	77 menit 55 detik
Distribusi aplikasi	2 menit 9 detik	3 menit 6 detik	2 menit 33 detik

Tabel 9. Proses kompilasi dan distribusi aplikasi dengan CI/CD pada environment production.

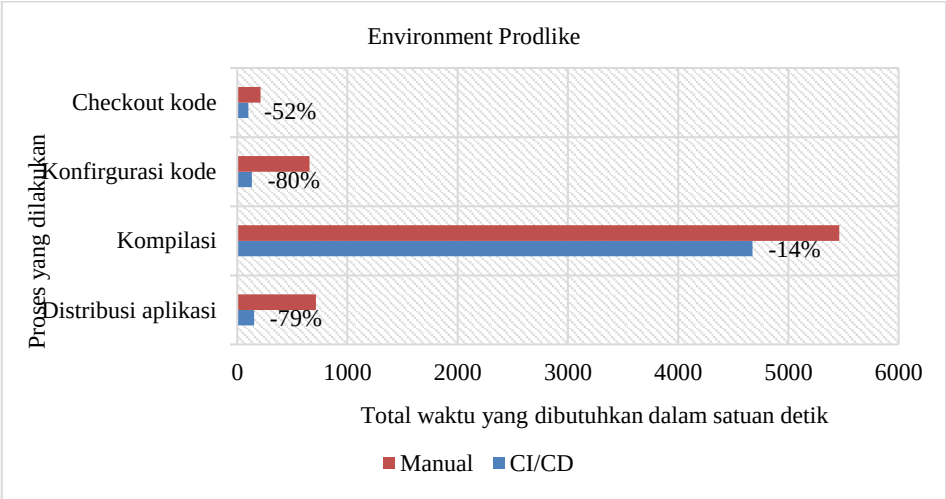
Proses yang dilakukan	Waktu yang dibutuhkan		
	Minimum	Maksimum	Rata-Rata
Checkout kode	1 menit 4 detik	2 menit 24 detik	1 menit 48 detik
Konfigurasi kode	42 detik	56 detik	48 detik
Kompilasi	52 menit 54 detik	55 menit 3 detik	53 menit 20 detik
Distribusi aplikasi	2 menit 33 detik	4 menit 43 detik	2 menit 48 detik

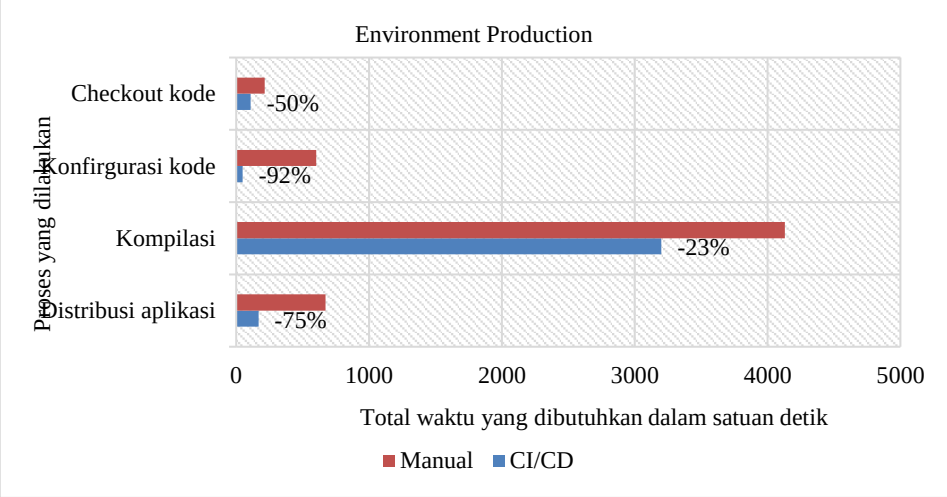
Informasi dari tabel 6, tabel 7, tabel 8, dan tabel 9 dapat digunakan untuk melakukan evaluasi efisiensi dan waktu yang diperlukan dalam setiap tahap, serta untuk memahami bagaimana CI/CD berdampak pada proses pengembangan aplikasi. Pada Gambar 6 dan Gambar 7 akan ditampilkan komparasi persentase efisiensi waktu antara proses manual dengan proses CI/CD.





Gambar 6. Grafik Perbandingan Waktu Antara Proses Manual dengan Proses CI/CD pada Environment Development dan Staging





Gambar 7. Grafik Perbandingan Waktu Antara Proses Manual dengan Proses CI/CD pada Environment Prodlike dan Production

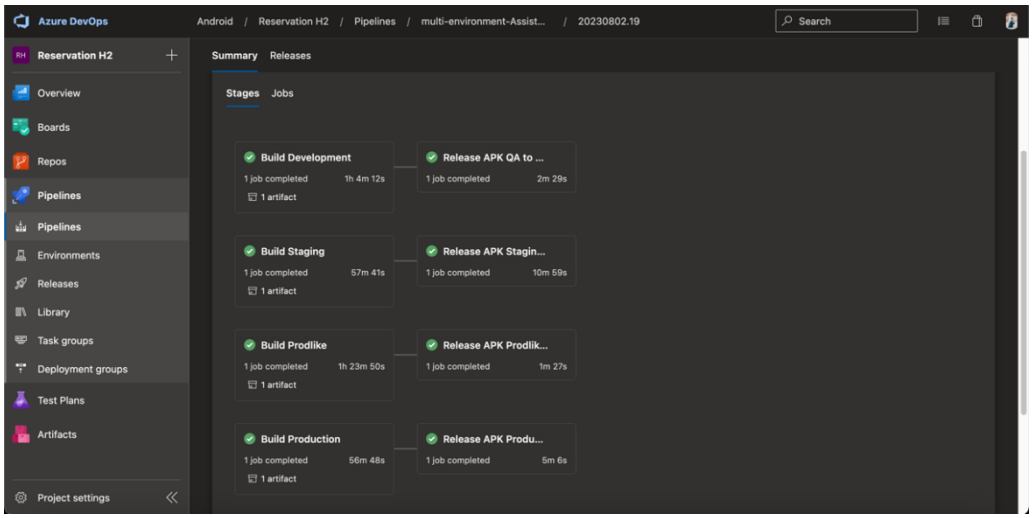
Dari gambar 6. dan gambar 7. dapat dilihat bahwa dampak penggunaan CI/CD dapat mengurangi waktu dari seluruh proses yang ada pada seluruh *environment*. Dampak pengurangan penggunaan waktu yang paling signifikan pada proses konfigurasi kode dimana waktu dapat dipangkas sekitar 90 persen. Kemudian waktu

pada proses checkout kode dapat dipangkas sebesar 50 persen. Dampak pengurangan penggunaan waktu paling sedikit terdapat pada proses kompilasi yang hanya berkurang sekitar 25 persen. Pada proses distribusi aplikasi waktu yang berkurang sekitar 75 persen.

Secara keseluruhan hasil dari proses manual apabila dilakukan secara berurutan adalah memakan waktu selama 6 jam, 41 menit, dan 13 detik. Sedangkan proses CI/CD apabila dilakukan secara berurutan akan memakan waktu selama 4 jam 28 menit 14 detik. Jumlah waktu yang berkurang apabila menggunakan CI/CD adalah sebesar 33 persen.

3.3 Kompilasi dan Distribusi Secara Paralel dengan CI/CD

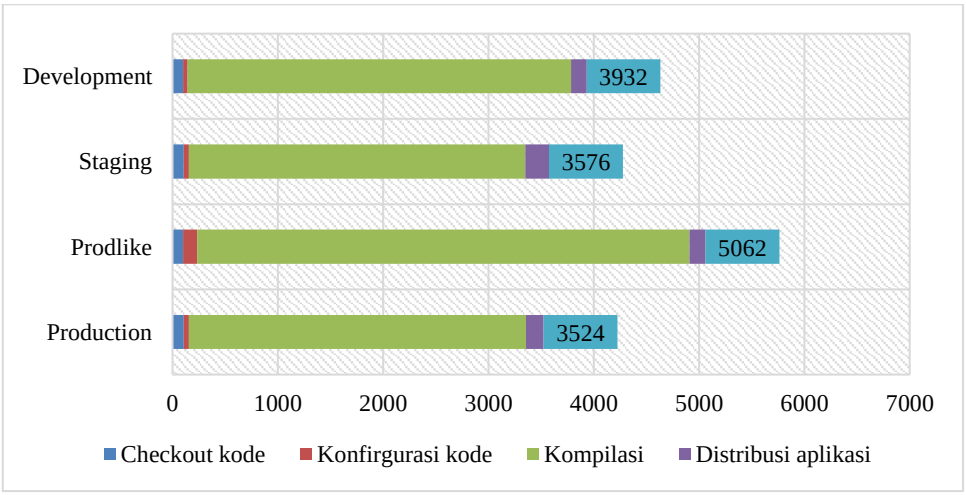
Pada sub-bab ini akan dibahas mengenai proses kompilasi dan distribusi aplikasi dengan menggunakan CI/CD secara paralel untuk seluruh *environment*, artinya proses CI/CD pada tiap-tiap *environment* akan dijalankan secara bersama-sama. Syarat untuk dapat menjalankan CI/CD secara paralel adalah environment aplikasi yang tersedia harus lebih dari satu, apabila aplikasi hanya memiliki satu buah environment maka metode CI/CD paralel ini tidak dapat dilakukan. CI/CD parallel ini akan berdampak pengurangan waktu yang lebih besar lagi dibandingkan dengan pengguakan CI/CD yang dilakukan secara berurutan. Tampilan user interface CI/CD paralel terdapat pada gambar 8.



Gambar 8. User Interface CI/CD Paralel

Pada gambar 8 diperlihatkan user interface CI/CD Paralel dimana bentuk *pipeline* nya berbeda dengan CI/CD berurutan di gambar 4. dan 5. Disini proses CD digabungkan menjadi satu pipeline dengan CI yang bergantung dari proses CI tiap *environment*-nya sendiri. CI setiap environment berjalan secara bersamaan, sedangkan CD dijalankan setelah CI selesai. Pada gambar 8 terlihat waktu yang digunakan untuk proses masing-masing CI dan masing-masing CD.

Berikut grafik presentase penggunaan CI/CD secara paralel.



Gambar 9. Grafik Waktu yang Dibutuhkan dalam Proses CI/CD Secara Paralel

Pada Gambar 9. ditampilkan jumlah total waktu yang digunakan untuk proses CI/CD pada tiap-tiap *environment*. Pada *environment* Development memakan waktu total sebanyak 3932 detik, *environment* Staging sebanyak 3576 detik, *environment* Prodlike sebanyak 5062 detik, dan *environment* Production sebanyak 3524 detik. Proses paralel ini dapat selesai keseluruhan pada saat *environment* terlama selesai yakni *environment* Prodlike yang selesai dalam 5062 detik atau selama 1 jam 24 menit 22 detik. Perbandingan hasil antara metode manual dengan CI/CD Berurutan serta dengan CI/CD Paralel ditampilkan pada Tabel 10.

Tabel 10. Perbandingan Waktu Tiap Metode

Metode	Waktu yang dibutuhkan	Persentase Selisih Waktu
Manual	6 jam, 41 menit, dan 13 detik	0
CI/CD Berurutan	4 jam 28 menit 14 detik	-33%
CI/CD Paralel	1 jam 24 menit 22 detik	-68%

Tabel 10. menunjukkan perbandingan waktu yang dibutuhkan untuk tiga metode pengujian yang berbeda: Manual, CI/CD Berurutan, dan CI/CD Paralel. Metode Manual memakan waktu paling lama, yaitu 6 jam, 41 menit, dan 13 detik. Sementara itu, CI/CD Berurutan dan CI/CD Paralel menunjukkan efisiensi yang signifikan dengan masing-masing waktu 4 jam 28 menit 14 detik dan 1 jam 24 menit 22 detik.

Dengan menerapkan CI/CD berurutan, terjadi peningkatan efisiensi waktu sebesar 33% dibandingkan dengan metode manual, sedangkan penggunaan CI/CD paralel menunjukkan peningkatan yang lebih besar, yakni sebesar 68% terhadap CI/CD berurutan dan 79% terhadap metode manual. Hasil ini mengindikasikan bahwa otomatisasi proses CI/CD, baik secara berurutan maupun paralel, memberikan dampak yang signifikan terhadap peningkatan efisiensi kompilasi dan distribusi aplikasi. CI/CD paralel memberikan keuntungan waktu yang lebih besar dibandingkan dengan CI/CD berurutan menunjukkan bahwa paralelisasi proses dapat menjadi pilihan yang sangat efektif dalam mengoptimalkan waktu kompilasi dan distribusi aplikasi. Meskipun CI/CD berurutan memberikan peningkatan yang signifikan, penggunaan CI/CD paralel menunjukkan potensi untuk mengurangi waktu kompilasi dan distribusi secara substansial. Oleh karena itu, dalam konteks pengembangan perangkat lunak, penggunaan CI/CD Paralel dapat dianggap sebagai solusi yang lebih menguntungkan untuk meningkatkan efisiensi kompilasi dan distribusi aplikasi. Pengurangan waktu yang substansial dapat membantu organisasi untuk lebih responsif terhadap perubahan pengembangan aplikasi.

Hasil penelitian menunjukkan bahwa implementasi CI/CD, baik dalam bentuk metode berurutan maupun metode paralel, memberikan keuntungan yang signifikan dalam hal efisiensi waktu. CI/CD paralel lebih menonjol sebagai pilihan yang lebih cepat, mengurangi waktu pengerjaan hingga 79% yang dapat memberikan dampak positif pada produktivitas tim pengembangan dibandingkan dengan metode berurutan yang hanya mengurangi waktu pengerjaan sebanyak 33%. Efisiensi ini dapat berdampak pada respons yang lebih cepat terhadap perubahan-perubahan kecil atau perbaikan aplikasi dapat diimplementasikan dan diuji dengan cepat, meminimalkan risiko kesalahan proses kompilasi dan distribusi aplikasi.

Peneliti juga melakukan perbandingan antar resource yang digunakan dengan mengubah variable RAM yang digunakan pada CI/CD berurutan dan CI/CD Paralel yang dapat dilihat pada tabel 11.

Tabel 11. Perbandingan Antar Resource Yang Digunakan

Spesifikasi	Proses CI/CD Berurutan	Proses CI/CD Berurutan	Proses CI/CD Paralel	Proses CI/CD Paralel
Edition	Windows Server 2016 Standard	Windows Server 2016 Standard	Windows Server 2016 Standard	Windows Server 2016 Standard
Processor	Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz 2.59 GHz	Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz 2.59 GHz	Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz 2.59 GHz	Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz 2.59 GHz
RAM	8.00 GB	16.00 GB	8.00 GB	16.00 GB
System type	64-bit operating system	64-bit operating system	64-bit operating system,	64-bit operating system
Hasil kompilasi dan distribusi	4 jam 28 menit 14 detik	3 jam 29 menit 43 detik	1 jam 24 menit 22 detik	54 menit 50 detik

Dapat dilihat pada tabel 11. bahwa spesifikasi RAM berpengaruh pada proses kompilasi dan distribusi aplikasi dimana RAM yang lebih besar akan mempercepat waktu yang digunakan.

4. KESIMPULAN

Kesimpulan dari penelitian ini sesuai dengan tujuan yang telah ditetapkan. Penelitian ini berhasil menjelaskan implementasi CI/CD pada proses kompilasi dan distribusi aplikasi mobile, sebagaimana yang diamanatkan oleh tujuan penelitian, yaitu untuk mengetahui perbandingan waktu yang didapatkan setelah

penerapan praktik CI/CD berurutan dan CI/CD paralel pada proses kompilasi serta distribusi aplikasi. Berdasarkan analisis dan perancangan yang dilakukan, dapat disimpulkan bahwa:

1. Penggunaan CI/CD secara berurutan mengurangi waktu pengerjaan sebesar 33% dibandingkan dengan metode manual, dari 6 jam 41 menit menjadi 4 jam 28 menit.
2. Penggunaan CI/CD secara paralel mengurangi waktu pengerjaan sebesar 68% dibandingkan dengan penggunaan CI/CD secara berurutan, dari 4 jam 28 menit menjadi 1 jam 24 menit.
3. Penggunaan CI/CD secara paralel juga mengurangi waktu pengerjaan sebesar 79% dibandingkan dengan metode manual, dari 6 jam 41 menit menjadi 1 jam 24 menit.

UCAPAN TERIMAKASIH

Kami dengan tulus ingin menyampaikan rasa terimakasih kepada semua pihak yang telah berpartisipasi dan memberikan kontribusi berharga dalam penelitian ini. Ucapan terima kasih kami sampaikan kepada seluruh individu dan lembaga yang telah memberikan bantuan serta dukungan yang luar biasa dalam memperlancar jalannya penelitian ini. Adapun kepada semua yang telah memberikan dukungan, nasihat, dan bantuan teknis selama proses penelitian, kami ingin mengucapkan rasa terima kasih yang sebesar-besarnya atas kontribusi yang berarti bagi kelancaran penyelesaian penelitian ini.

REFERENSI

[1] Tohirin, S. F. Utam, S. R. Widianto dan W. A. Mauludyansah, “Implementasi DevOps pada Pengembangan Aplikasi e-Skrining Covid-19,” 2020.

[2] Jaeni, A. D. L. dan N. A. S., “Implementasi Continuous Integration / Continuous Delivery (CI/CD) Pada Performance Testing Devops,” 2022.

[3] K.Janani, K.Anuhya, V.L.Manaswini, V.Likitha, B.Suneetha dan T.Vignesh, “Analysis of CI/CD Application in Kubernetes Architecture,” 2022.

[4] I.-C. Donca, O. P. Stan, M. Misaros, D. Gota dan L. Miclea, “Method for continuous integration and deployment using a pipeline generator for agile software projects,” 2022.

[5] Alexander, “Rancang Bangun Sistem Continuous Integration Pipeline Menggunakan Jenkins Di PT. Emporia Digital Raya,” 2022.

[6] M. Abdurrahman, “Implementasi Sistem Continuous Integration Dan Continuous Deployment (CI/CD) Pada Proyek Perangkat Lunak Mahasiswa,” *(Doctoral dissertation, Universitas Gadjah Mada),* 2020.

[7] L. N. Ba, “Enterprise-grade CI/CD Pipeline For Mobile Development,” 2022.

[8] I. Karamitsos, S. Albarhami dan C. Apostolopoulos, “Applying DevOps Practices of Continuous Automation for Machine Learning,” 2020.

[9] R. Wijaya, A. Prasetyo dan D. Wahyuningsih, “Implementasi CI/CD Untuk Build dan Deploy Website Dengan Docker Runner Pada Organisasi Belajar Linux ID,” 2021.

[10] B. Permadi, “DevSecOps Support on Continuous Integration Deployment of TRAC Applications for Mobile iOS and Android with Continuous Integration Method,” 2021.

[11] D. Wijayanto, A. Firdonsyah dan F. D. Adhinata, “Implementasi Continous Integration / Continous Delivery Menggunakan Process Manager 2,” 2021.

[12] A. Alpery dan M. A. F. Ridha, “Implementasi Ci/cd Dalam Pengembangan Aplikasi Web Menggunakan Docker Dan Jenkins,” 2021.

[13] R. Adrian, A. K. Fauziyyah dan S. Alam, “Optimasi Kecepatan Continuous Integration/ Continuous Delivery pada Otomatisasi Jaringan berbasis Grey Wolf Optimizer,” 2022.

[14] D. Enda, Supria, I. Yulia, M. F. Amirul dan M. F. Asyrofi, “Penerapan Continuous Integration(CI) Pada Aplikasi Web Profil Karang Taruna (Studi Kasus : Karang Taruna Kabupaten Bengkalis),” 2022.

[15] I. G. Noviantama dan A. P. Wahyu, “Implementasi Contionous Integration Dan Continous Deployment Pada Aplikasi Learning Management System Di PT. Millennia Solusi Informatika,” 2021.

[16] H. E. Wahanani, W. S. J. Saputra dan B. H. V. Wahono, “Perancangan Infrastruktur Server VCS (Version Control System) Dengan Gitlab Berbasis Git,” 2019.

[17] H. Toba, T. K. Gautama, J. Narabel, A. Widjaja dan S. F. Sujadi, “Evaluasi Metodologi CI/CD untuk Pengembangan Perangkat Lunak dalam Perkuliahan,” 2022.

[18] A. M. Shama dan D. W. Chandra, “Implementasi Static Application Security Testing Menggunakan Jenkins CI/CD Berbasis Docker Container Pada PT. Emporia Digital Raya,” 2021.

[19] A. D. Pertiwi, “Sistem Informasi Jasa Laundry Pada Laundry Denok Berbasis Web Menggunakan Metode System Development Life Cycle (SDLC),” 2020.

[20] M. Ridwan, I. Fitri dan Benrahman, "Rancang Bangun Marketplace Berbasis Website menggunakan Metodologi Systems Development Life Cycle (SDLC) dengan Model Waterfall," 2021.

[21] R. R. Illahi, N. N. Anwari dan A. Primajaya, "Tingkat Keefektifan Pengembangan Sistem Informasi Dalam Era Revolusi Industri 4.0," 2022.

[22] A. Farid dan I. G. Anugrah, "Implementasi CI/CD Pipeline Pada Framework Androbase Menggunakan Jenkins (Studi Kasus: PT. Andromedia)," 2021.

[23] T. N. Fajarotun, "Analisa Dan Pengembangan Private Cloud Computing Berbasis Infrastructure As A Service (IAAS) Pada Sman 1 Metro," 2020.

[24] Y. F. Aladina, A. Bhawiyuga, R. A. Siregar dan P. H. Trisnawan, "Penerapan Mekanisme Continuous Deployment dalam Pengembangan dan Pembaruan Perangkat Lunak Sistem Benam Berbasis Internet of Things," *Jurnal Teknologi Informasi dan Ilmu Komputer,* 2022.

[25] T. Rangnau, R. v. Buijtenen, F. Fransen dan F. Turkmen, "Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines," *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC),* pp. 145-154, 2020.

[26] R. A. Parama, H. Studiawan dan R. J. Akbar, "IMPLEMENTASI CONTINUOUS INTEGRATION DAN CONTINUOUS DELIVERY PADA APLIKASI MYITS SINGLE SIGN ON," *Jurnal Teknik ITS,* pp. A264-A269, 2022.

[27] L. Andraini, "Pengimplementasian Devops Pada Sistem Tertanam Dengan ESP8266 Menggunakan Mekanisme Over The Air," 2022.

[28] V. Dakic, J. Redzepagic dan M. Basic, "CI/CD toolset security," *Proceedings of the 32nd DAAAM International Symposium, ISSN,* pp. 1726-9679, 2022.

[29] R. Agrawal dan N. Pandey, "Strategies for Developing and Deploying Enterprise-Level Mobile Applications on a Large Scale: A Comprehensive Analysis," *International Journal of Enhanced Research in Management & Computer Applications,* 2020.

[30] J. Maytanius, Leonardo, H. Wahyudi, Kelvin, R. Lim dan Benny, "Analisis Manajemen Operasional Perusahaan Astra International," *Jurnal Pengabdian kepada Masyarakat Nusantara (JPkMN) ,* vol. 4, no. 2, pp. 1553-1557, 2023.