**Jurnal Indonesia : Manajemen Informatika dan Komunikasi**
https://journal.stmiki.ac.id

E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May

OPEN ACCESS

https://doi.org/10.35870/jimik.v5i2.832

# Implementasi *Monitoring* Sistem Perusahaan *On-Premises* dan *Cloud* Menggunakan Teknologi Jenkins

Sinatria Banyu Adil [1], Yos Richard Beeh [2*]

[1,2*] Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Kota Salatiga, Provinsi Jawa Tengah, Indonesia.

*Email*: 672020166@student.uksw.edu [1], yosrichardbeeh@uksw.edu [2*]

**Histori Artikel:**
*Dikirim* 19 April 2024; *Diterima dalam bentuk revisi* 26 April 2024; *Diterima* 10 Mei 2024; *Diterbitkan* 20 Mei 2024.


## Abstrak

Penelitian ini mengimplementasikan Jenkins untuk memantau dan mengelola proses job, serta integrasinya dengan infrastruktur on-premises dan cloud di PT. AAA. Compute Engine digunakan sebagai server utama untuk menjalankan Jenkins. Tahapan meliputi konfigurasi server Compute Engine, instalasi plugin penting, dan penggunaan library Python. Hasilnya menunjukkan Jenkins berhasil dalam menjalankan proses job antara sistem on-premises dan cloud. Email Extension Plugin digunakan untuk memantau proses job dan memberikan notifikasi ke PIC jika terjadi kesalahan. Pengelolaan akses diterapkan melalui Role-based Authorization Strategy. Pengujian dilakukan menggunakan blackbox testing dengan tingkat keberhasilan 100%. Diharapkan implementasi ini dapat meningkatkan efisiensi dan efektivitas manajemen proses job di PT. AAA.

**Kata Kunci:** Jenkins; Pemantauan Job; Manajemen Job; Infrastruktur On-Premises; Infrastruktur Cloud.

## Abstract

This research implements Jenkins for monitoring and managing job processes, as well as its integration with on-premises and cloud infrastructures at PT. AAA. Compute Engine is used as the main server to run Jenkins. The stages include configuring the Compute Engine server, installing important plugins, and using Python libraries. The results show that Jenkins successfully runs job processes between on-premises and cloud systems. The Email Extension Plugin is used to monitor job processes and send notifications to PICs in case of errors. Access management is implemented through Role-based Authorization Strategy. Testing is done using blackbox testing with a success rate of 100%. It is hoped that this implementation can improve the efficiency and effectiveness of job process management at PT. AAA.

**Keyword:** Jenkins; Job Monitoring; Job Management; On-Premises Infrastructure; Cloud Infrastructure.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.832

# 1. Pendahuluan

PT. AAA, yang mulai beroperasi sebagai gerai pada tahun 1989, telah berkembang menjadi salah satu jaringan toko swalayan terbesar di Indonesia. Perusahaan ini tidak hanya sukses di dalam negeri, tetapi juga telah memperluas bisnisnya ke Filipina pada tahun 2014. Pada tahun 2019, PT. AAA memiliki lebih dari 14.300 toko yang beroperasi, dengan lebih dari 126.589 karyawan yang bekerja di satu kantor pusat dan 32 kantor cabang (Alfamart, n.d.), (Alfamartku, n.d.). Dalam menjalankan operasionalnya, PT. AAA mengandalkan infrastruktur IT yang kompleks, termasuk penggunaan banyak *server* dan proses *job* yang mendukung kegiatan operasional, yang dikelola melalui Crontab di lingkungan Linux. Dalam operasional perusahaan, pengelolaan sistem IT memegang peran yang sangat vital dalam memastikan kinerja, ketersediaan, dan keamanan yang efisien. Perusahaan retail besar seperti PT. AAA sering menghadapi tantangan dalam pengelolaan infrastruktur *on-premises* yang melibatkan banyak *server* dan proses *job* yang mendukung operasi mereka, menggunakan Crontab dalam lingkungan Linux. Sebagaimana dilaporkan oleh *IndonesianCloud* pada tanggal 22 September 2023, '*Cloud computing* telah menjadi tren utama dalam beberapa tahun terakhir' dan menurut laporan dari *Gartner* pada tahun 2020, 'Sekitar 72% perusahaan di seluruh dunia telah mengadopsi teknologi *cloud* untuk keperluan bisnis mereka.'. Dengan pertumbuhan data dan aplikasi yang terus berkembang, integrasi antara sistem *on-premises* dan *cloud* menjadi penting untuk mengoptimalkan sumber daya, memanfaatkan skalabilitas, dan menjaga keamanan data kunci di lingkungan *on-premises*. Menurut Jain, A. (2020). *Monitoring and Deployment in Cloud Infrastructure*. Dalam penelitiannya, Jain membahas tentang Jenkins, sebuah alat otomatisasi sumber terbuka yang ditulis dalam bahasa Java dan digunakan untuk tujuan integrasi berkelanjutan. Jenkins memungkinkan pengembang untuk secara terus-menerus membangun dan menguji proyek perangkat lunak, memfasilitasi integrasi perubahan ke dalam proyek dengan mudah. Dengan kehadiran *plugin*, Jenkins menyediakan integrasi berkelanjutan dengan berbagai alat, termasuk *HTML Publisher* dan Git. Keuntungan utama Jenkins mencakup dukungan komunitas yang besar, kemudahan instalasi, ketersediaan banyak *plugin*, biaya nol, dan kemampuan penggunaan lintas *platform* karena dikembangkan dalam bahasa Java. Proses integrasi berkelanjutan yang dilakukan oleh Jenkins melibatkan pemantauan teratur terhadap repositori bersama, pendeteksian perubahan, pembangunan proyek, notifikasi tim terkait jika proses pembangunan gagal, instalasi pada *server* uji, pengujian, pembuatan laporan, dan pemberitahuan kepada pengembang. Keseluruhan, Jenkins mempercepat proses pengembangan dan pengujian melalui otomatisasi yang efisien.

Menurut Golec, D., Strugar, I., & Belak, D. (2021). *The Benefits of Enterprise Data Warehouse Implementation in Cloud vs. On-premises*. Dalam *paper* tersebut, dijelaskan manfaat implementasi data *warehouse* perusahaan di *cloud* dibandingkan dengan *on-premises*. Terdapat kecenderungan umum dalam pengelolaan *data warehouse* menuju *cloud*, yang dapat memberikan keunggulan kompetitif yang signifikan jika dilakukan dengan baik. Kedua model tersebut memiliki sejumlah keuntungan, dan strategi organisasi berdasarkan kurva kematangan analitis dapat menunjukkan model gudang data mana yang lebih sesuai untuk implementasi perusahaan. Sebagai alternatif dari *over provisioning* untuk mengatasi lonjakan beban tertinggi, perusahaan dapat mengatur kapasitas dan mengaktifkan instansi gudang data secara fleksibel sesuai dengan kebutuhan. Isu yang timbul di lingkungan *cloud* memiliki relevansi yang signifikan, mendorong perusahaan untuk memilih jenis gudang data perusahaan yang bersifat campuran atau hibrida. Menurut Rahman, D., Amnur, H., & Rahmayuni, I. (2020). *Monitoring Server* dengan Prometheus dan Grafana serta Notifikasi Telegram. *Monitoring server* berhasil dilakukan melalui penggunaan Prometheus dan Grafana dengan hasil yang memuaskan. Sistem operasi yang digunakan pada *server monitoring* adalah Ubuntu Server versi 18.04. Sistem ini memberikan notifikasi kepada administrator jika terjadi permasalahan pada CPU, memori, atau layanan Apache dan MySQL. Grafana akan mengirimkan pemberitahuan melalui Telegram apabila kondisi *server* melampaui batas yang telah ditetapkan atau jika layanan Apache maupun MySQL mengalami gangguan. Pemilihan *flavor* didasarkan pada spesifikasi VCPU, RAM, dan Disk sesuai dengan spesifikasi minimal untuk membuat *instance*. Sebagai pengembangan berikutnya, diharapkan sistem *monitoring server* dapat

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.832

memberikan pemberitahuan ke aplikasi *mobile* yang telah dibuat. Implementasi *monitoring server* dengan Prometheus dan Grafana memiliki potensi pengembangan yang lebih lanjut, terutama untuk memantau layanan lain seperti SSH, *mail*, Jenkins, Kubernetes, dan sebagainya. Menurut Kerkelä, J. (2018). *Continuous Integration Server Performance Monitoring*. Dalam tesis ini, penelitian dilakukan pada proyek integrasi berkelanjutan dengan proyek pengembangan berskala besar sebagai proyek uji coba. Masalah yang diteliti adalah variasi durasi pembangunan dalam uji coba otomatis di CI *server*, yang menggunakan *server* Jenkins. Variasi durasi pembangunan menyebabkan tingkat kinerja layanan yang fluktuatif bagi pengembang. Tujuan penelitian ini adalah mengimplementasikan artefak yang memberikan informasi tentang kinerja *server* Integrasi Berkelanjutan dari waktu ke waktu dengan mengimplementasikan alat pemantauan pada CI *server*. Tujuannya adalah mendeteksi peristiwa variasi kinerja dan mengidentifikasi penyebab utama variasi durasi pembangunan dari informasi kinerja. Alat pemantauan, diimplementasikan sebagai *plugin* Jenkins, berhasil memenuhi persyaratan proyek CI dan proyek pelanggan. Pengujian awal dilakukan pada *server* uji, dan alat pemantauan berhasil digunakan dalam lingkungan Jenkins proyek pelanggan. Alat pemantauan memberikan data kinerja, khususnya dari *executor* individu yang digunakan untuk pembangunan di *server* Jenkins, dan menggabungkan data kinerja dengan data pembangunan dari pembangunan tertentu yang sedang diuji. Dengan fungsionalitas ini, alat pemantauan memberikan solusi untuk masalah penelitian.

Jenkins adalah *platform open-source* yang digunakan untuk otomatisasi proses pengembangan perangkat lunak. Dalam sistem ini, Jenkins memiliki fitur utama yang mendukung *Continuous Integration* (CI) untuk mengintegrasikan kode ke repositori secara terus-menerus. Jenkins juga mendukung otomatisasi pengujian dengan menjalankan serangkaian tes otomatis setiap kali ada perubahan kode, serta *Continuous Deployment* (CD) untuk mengirimkan perubahan ke lingkungan produksi setelah melewati pengujian (Armenise, 2015). Dalam penelitian ini, Jenkins dimanfaatkan untuk tujuan menggabungkan berbagai *jobs* yang berjalan terus menerus dari berbagai *server* yang berbeda menjadi satu *server* Jenkins. Hal ini dilakukan untuk memudahkan *monitoring* dan manajemen semua *jobs* dari satu tempat. Selain itu, jika terjadi *error* dalam proses *job*, Jenkins akan mengirimkan *email* notifikasi kepada pengguna terkait. Untuk mendukung fungsionalitas ini, penelitian menggunakan beberapa *plugin*, seperti *Pipeline Utility Steps* menyediakan sejumlah langkah tambahan yang bisa digunakan dalam Jenkinsfile untuk melakukan berbagai tugas, termasuk manipulasi *string*, pengaturan variabel lingkungan (*environment variable*), dan lainnya, *Email Extension Plugin* untuk mengirim email notifikasi, *PostgreSQL Database Plugin* untuk integrasi dengan *database PostgreSQL*, serta *Role-based Authorization Strategy* untuk mengatur hak akses pengguna sesuai dengan peran masing-masing. Dengan demikian, Jenkins tidak hanya berperan dalam integrasi kode dan otomatisasi pengujian, tetapi juga dalam manajemen keseluruhan proses pengembangan perangkat lunak (Antunes *et al.*, 2018). Pemantauan proses job sangat penting dalam manajemen infrastruktur karena membantu meningkatkan kinerja dan mengurangi *downtime*. Dalam penelitian ini, pemantauan ini diperlukan untuk mengatasi permasalahan yang dihadapi. Untuk itu, beberapa teknik dan alat pemantauan, seperti l*ogging, monitoring real-time, alerts, monitoring dashboard*, dan analisis *log*, digunakan untuk memberikan visibilitas dan pemahaman yang komprehensif terhadap proses *job*. Dengan memahami dan menerapkan pemantauan proses *job* secara efektif, konsep ini dapat digunakan dalam penelitian ini untuk memberikan keterampilan dalam menangani potensi masalah dengan cepat dan proaktif, sehingga dampak negatif terhadap kinerja perusahaan dapat diminimalkan (PT. Artajasa Pembayaran Elektronis, 2021).

Selain memperhatikan aspek *monitoring,* penelitian ini juga menekankan pentingnya integrasi antara *database* PostgreSQL (*cloud*) dan Oracle (*on-premises*). Integrasi ini memungkinkan pertukaran data antara kedua *database*, yang penting untuk menjaga konsistensi data dan meningkatkan efisiensi operasional. Integrasi sistem *on-premises* dan *cloud* memiliki peran penting dalam mendukung efisiensi operasional perusahaan, karena menggabungkan keunggulan infrastruktur *on-premises* dan fleksibilitas *cloud computing* untuk optimalisasi sumber daya, meningkatkan skalabilitas, dan mencapai efisiensi biaya. Meskipun penting, integrasi ini juga menimbulkan tantangan dalam menyelaraskan teknologi yang berbeda, menjaga tingkat keamanan yang konsisten, dan mengelola infrastruktur secara

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.832

terintegrasi. Namun, integrasi ini membawa manfaat seperti fleksibilitas dalam penyimpanan data, efisiensi biaya, responsivitas terhadap perubahan bisnis, dan kemampuan untuk mengadopsi inovasi teknologi dengan lebih cepat (Cloudeka, 2023). Python ialah bahasa pemrograman yang memiliki keunggulan yang terletak pada kombinasi kapabilitas, sintaksis kode yang singkat dan jelas, serta ketersediaan pustaka fungsionalitas yang luas dan banyak, membuatnya menjadi pilihan yang tepat untuk penelitian ini (Saragih, 2018). Dalam penelitian ini, Python akan digunakan untuk membangun skrip yang akan dijalankan di Jenkins. Skrip ini akan bertugas untuk memantau dan mengotomatisasi *job* sinkronisasi antara *database on-premises* dan *cloud*. Python cx_Oracle *library* akan digunakan untuk menghubungkan Python dengan *database* Oracle dan Python psycopg2 *library* akan digunakan untuk menghubungkan Python dengan *database* PostgreSQL. Python juga dapat digunakan untuk menarik data dari situs web, berinteraksi dengan API guna otomatisasi pengambilan data, dan mengintegrasikan dengan *web services* (Narayani *et al*., 2022).

Penelitian ini memanfaatkan *database* PostgreSQL yang mendukung untuk terhubung dengan infrastruktur *cloud computing*, sehingga sesuai dengan penggunaan Google Cloud Platform dalam penelitian ini. Dengan fitur-fitur tersebut, RDBMS ini, yang dikembangkan oleh Departemen Ilmu Komputer Berkeley, memberikan kemudahan bagi pengguna dalam mengimplementasikan sistem ini dan dapat memenuhi kebutuhan proses aplikasi data yang juga mendukung SQL dan ACID-compliant, yang berarti bahwa transaksi *database* dapat diproses dengan konsisten, bahkan dihadapkan dengan *software/hardware failures* (Yuliardi., n.d.), (Eessaar, 2021). Oracle merupakan salah satu Sistem Manajemen Basis Data Relasional (RDBMS) yang dikembangkan oleh Oracle Corporation. *Database* ini umumnya digunakan oleh banyak perusahaan yang menggunakan infrastruktur *on-premises*. Dalam penelitian ini, Oracle *Database* dengan dibantu *SQL*Plus* dan *Instant Client Oracle* digunakan untuk mengembangkan sistem yang beroperasi pada infrastruktur *on-premises* (Oracle, 2021). Google Cloud Platform adalah *platform cloud computing* yang disediakan oleh Google. *Platform* ini menyediakan berbagai layanan, seperti Cloud SQL, Cloud Run, Compute Engine, dan lain-lain. Compute Engine menyediakan *virtual machine* yang dapat disesuaikan dengan berbagai sistem operasi, perangkat lunak, dan konfigurasi perangkat keras sesuai keinginan pengguna yang dimana ini sesuai dengan kebutuhan dari penelitian ini. Pengguna dapat memilih tipe mesin yang telah dikonfigurasi sesuai kebutuhan (Google, 2021), (Mufti *et al*., 2021). Compute Engine juga menyediakan fitur-fitur seperti *load balancing, auto-scaling,* dan berbagai alat pemantauan untuk membantu pengguna mengelola *virtual machine* mereka (Ahuja, 2020).

Berdasarkan hasil wawancara dengan salah satu tim *IT Development* di PT. AAA, ditemukan bahwa *Head Office* PT. AAA mengoperasikan beberapa *server* yang tersebar di seluruh infrastruktur perusahaan. Setiap *server* bertugas menjalankan berbagai proses *job* yang mendukung operasional perusahaan. Namun, keberadaan banyak *server* ini menimbulkan beberapa permasalahan, terutama dalam hal pemantauan dan manajemen proses *job*. Salah satu isu utama yang dihadapi adalah kesulitan dalam mendeteksi ketika suatu *job* atau proses berhenti beroperasi tanpa pemberitahuan dari pengguna (PIC). Dalam penelitian ini, PT. AAA perlu mencari solusi untuk membantu karyawan mengatasi setiap masalah yang muncul. Tantangan ini menjadi serius karena tim IT kesulitan merespons secara efektif ketika masalah timbul, baik karena masalah teknis maupun kesalahan manusia. Untuk mengatasi masalah ini, perlu untuk mengkonsolidasikan proses *job* yang awalnya terdistribusi pada beberapa *server* menjadi satu *server* sentral. *Server* ini akan dilengkapi dengan aplikasi pemantauan yang menggunakan Teknologi Jenkins. Dikutip dari *Simplilearn* pada tanggal 22 Mei 2023, 'Jenkins berfungsi mengawasi sistem kontrol versi dan memulai serta memonitor sistem pembangunan jika ada perubahan yang terjadi. Jenkins secara cermat mengawasi keseluruhan proses ini dan memberikan laporan serta pemberitahuan untuk memberikan peringatan'. Salah satu fitur tersebut relevan dengan kebutuhan untuk mendeteksi proses yang mati dan memberikan notifikasi otomatis kepada PIC yang terkait. Penelitian ini bertujuan untuk mengatasi masalah yang dihadapi oleh PT. AAA dalam mengelola infrastruktur *on-premises* dan mengintegrasikannya dengan teknologi *cloud*. Dengan mengimplementasikan solusi berbasis Teknologi Jenkins, diharapkan perusahaan dapat lebih efisien dalam mengelola dan memantau proses pekerjaan, sehingga operasionalnya berjalan

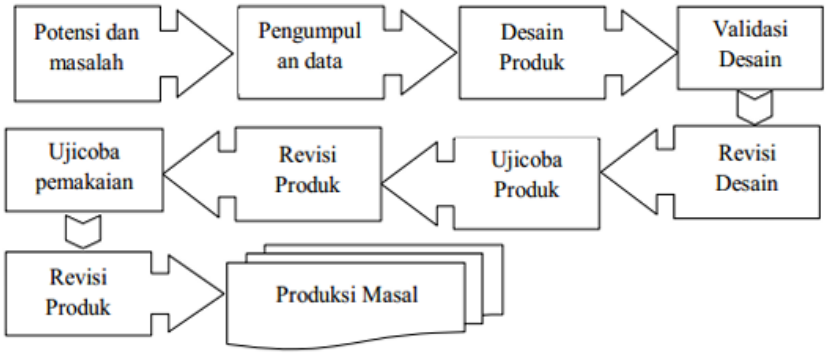
Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.832

dengan lancar. Selain itu, penelitian ini juga diharapkan dapat memberikan panduan praktis kepada organisasi lain yang menghadapi tantangan serupa dalam mengelola sistem *on-premises* dan *cloud computing*. Dengan demikian, langkah-langkah ini diharapkan dapat membawa manfaat yang signifikan dalam mendukung kemajuan dan efektivitas operasional perusahaan, sejalan dengan tren perkembangan teknologi informasi yang semakin penting di era bisnis modern.

Penelitian ini dilatarbelakangi oleh kebutuhan PT. AAA, sebuah perusahaan retail besar dengan jaringan toko yang luas, untuk mengatasi tantangan dalam mengelola infrastruktur *on-premises* yang melibatkan banyak *server* dan proses *job*. Dengan adopsi teknologi *cloud* yang semakin meningkat, integrasi antara sistem *on-premises* dan *cloud* menjadi krusial untuk mengoptimalkan sumber daya dan menjaga keamanan data. Penelitian ini berusaha menjawab pertanyaan tentang "Bagaimana penerapan Jenkins dalam pemantauan dan otomatisasi proses *job* sinkronisasi antara *database on-premises* dan *cloud* ataupun sebaliknya dapat dilakukan dengan baik?". Solusi ini diharapkan dapat meningkatkan efisiensi PT. AAA dalam mengelola operasional perusahaannya. Selain bermanfaat bagi PT. AAA, hasil penelitian ini juga diharapkan dapat memberikan panduan praktis bagi organisasi lain yang menghadapi tantangan serupa dalam mengelola sistem *on-premises* dan *cloud computing*.

## 2. Metode Penelitian

Penelitian ini menggunakan metode *Research and Development* (R&D) untuk mengembangkan aplikasi *monitoring* sistem perusahaan berbasis teknologi Jenkins. Metode R&D dipilih karena sesuai dengan tujuan penelitian yang ingin mengembangkan produk baru berupa aplikasi tersebut. Penggunaan metode penelitian *Research and Development* (R&D), yang dipilih karena sesuai dengan tujuan penelitian yang ingin mengembangkan suatu produk atau teknologi baru. Menurut Sugiyono (2011), metode R&D merupakan metode penelitian yang dilakukan secara sistematis dan terstruktur, dengan tujuan yang jelas untuk mengembangkan produk atau teknologi baru, serta melibatkan pengujian dan evaluasi produk atau teknologi yang dikembangkan (Sugiyono, 2013).

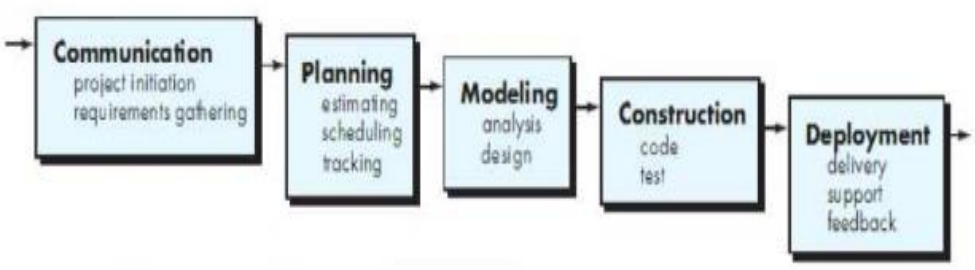



Gambar 1. Tahapan R&D

Tahapan dalam metode R&D menurut Sugiyono (2013) terdiri dari sepuluh tahapan yaitu potensi dan masalah, pengumpulan data, desain produk, validasi desain, uji coba pemakaian, revisi produk, uji coba produk, revisi desain, revisi produk, dan produksi massal. Pada penelitian ini, tahapan yang dilakukan meliputi: 1) Potensi dan Masalah: Identifikasi masalah monitoring sistem di PT. AAA yang menggunakan Crontab dalam lingkungan Linux, menghadapi kesulitan mendeteksi job atau proses yang berhenti tanpa pemberitahuan, dan memerlukan adopsi teknologi *Cloud*; 2) Pengumpulan Data: Mengumpulkan informasi mengenai infrastruktur yang ada dan tantangan monitoring di *On-Premises* dan *Cloud*; 3) Desain Produk: Merancang aplikasi monitoring dengan Jenkins, menggunakan plugin seperti *Pipeline Utility Steps*, *Email Extension*, *PostgreSQL Database*, dan *Role-based Authorization Strategy*; 4) Validasi Desain: Memvalidasi desain aplikasi dengan uji coba prototipe ke Supervisor dan Team

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.832

IT PT. AAA untuk memastikan efektivitas desain; 5) Perbaikan Desain: Menyesuaikan desain berdasarkan umpan balik validasi untuk meningkatkan kualitas dan kinerja; 6) Uji Coba Produk: Menguji aplikasi dengan data simulasi untuk memastikan fungsionalitas dan optimasi di berbagai beban kerja; 7) Revisi Produk: Memodifikasi aplikasi berdasarkan hasil uji coba untuk meningkatkan akurasi dan mengurangi kesalahan deteksi; 8) Uji Coba Pemakaian: Melakukan uji coba pemakaian oleh pengguna beta dari Team IT PT. AAA dalam lingkungan produksi dan mengumpulkan umpan balik; 9) Revisi Produk: Mengupdate aplikasi berdasarkan umpan balik uji coba pemakaian dengan menambahkan fitur tambahan untuk meningkatkan fungsionalitas. Pembuatan Produk Masal: Aplikasi *monitoring* sistem diproduksi secara massal untuk implementasi di lingkungan produksi PT. AAA setelah aplikasi telah melewati semua tahapan pengembangan dan evaluasi dengan baik.




Gambar 2. Tahapan *Waterfall*

Untuk pengembangan aplikasi *monitoring* sistem, penelitian ini menggunakan model pengembangan *Waterfall* menurut referensi Pressman (2015) yang digambarkan tahapannya seperti pada Gambar 2 (Pressman, 2015) Model ini dipilih karena memiliki tahapan yang terstruktur dan jelas, cocok untuk proyek dengan spesifikasi yang telah ditentukan seperti dalam penelitian ini. Royce (1970) dan Boehm (1988) menyebutkan bahwa *Waterfall* adalah model yang mudah dipahami dan diimplementasikan, cocok untuk proyek dengan spesifikasi yang jelas dan stabil (Royce, 1970);(Boehm, 1988). Namun, Pressman (2010) mengingatkan bahwa *Waterfall* dapat menjadi kaku dan sulit untuk beradaptasi dengan perubahan (Pressman, 2010). Dalam penelitian ini, model *Waterfall* diadaptasi dengan beberapa penyesuaian untuk mengakomodasi kebutuhan proyek dan potensi perubahan.

Tahap pertama adalah *Communication* (*Project Initiation & Requirements Gathering*), di mana dilakukan identifikasi permasalahan fokus penelitian terkait solusi pemantauan sistem PT. AAA menggunakan Jenkins, dengan mengumpulkan informasi dari berbagai literatur terkait monitoring sistem, otomatisasi CI/CD, dan teknologi Jenkins. Tahap kedua adalah *Planning*, yang mencakup pengumpulan data hasil wawancara dengan tim IT dan data internal perusahaan, serta perencanaan pengembangan aplikasi monitoring sistem termasuk langkah-langkah pengembangan, alokasi sumber daya, estimasi waktu, dan penjadwalan proyek secara keseluruhan. Pada tahap *Modeling*, dilakukan analisis mendalam dan perancangan pemodelan aplikasi, mencakup pemodelan proses, pemodelan data, dan desain *output email* untuk memastikan alur kerja sistem dan struktur data yang efisien serta notifikasi yang informatif. Selanjutnya, pada tahap *Construction*, tim mempersiapkan infrastruktur, menginstal perangkat lunak, membuat *file job*, mengkonfigurasi Jenkins, dan melakukan uji coba untuk memastikan eksekusi job berjalan sesuai harapan. Tahap terakhir adalah *Deployment*, yang melibatkan implementasi otomatisasi, pengaturan auto-email untuk notifikasi kegagalan proses build, integrasi dengan mitra perusahaan, *monitoring* dan *feedback*, serta uji coba dan analisis oleh tim *Quality Assurance* (QA) perusahaan. Setelah semua tahapan tersebut dilalui dan standar terpenuhi, dilakukan implementasi aplikasi di PT. AAA.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.832

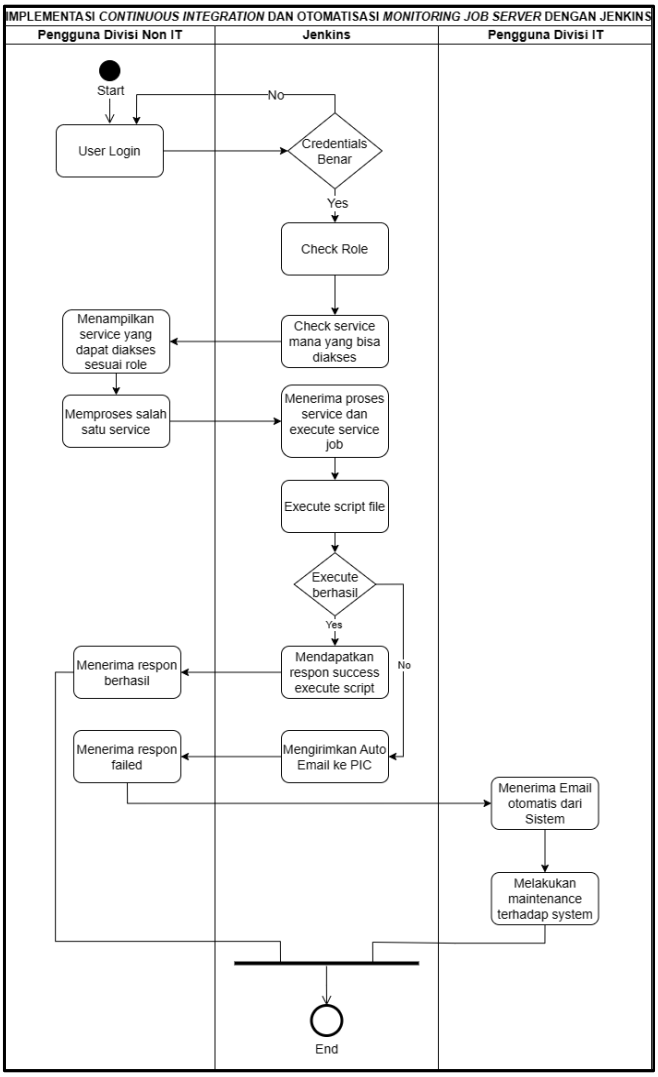
## 3. Hasil dan Pembahasan

Hasil dari penelitian *monitoring* ini, bisa dilihat dari implementasi Jenkins untuk pemantauan dan manajemen *job*, sekaligus integrasinya dengan infrastruktur *on-premises* dan juga *cloud*. Sebelum pada tahapan implementasi, penulis merancang sebuah *use case diagram,* dan juga *activity diagram*.


Gambar 3. *Use Case Diagram*

Gambar 3 adalah diagram *use case* yang menggambarkan dan menjelaskan tentang hak akses dan batasan dari peran *user* IT dan Non-IT dalam penggunaan Jenkins. *Use case* diagram ini memberikan gambaran tentang bagaimana setiap peran *user* dapat berinteraksi dengan Jenkins dalam lingkungan infrastruktur yang terintegrasi antara *server on-premises* dan *cloud*. *User* IT dapat memiliki akses penuh untuk mengelola dan mengatur konfigurasi Jenkins. Sementara *user* Non-IT memiliki akses terbatas untuk melihat status *job* dan hasil pemantauan. Hal ini membantu dalam memastikan keamanan dan keteraturan dalam penggunaan Jenkins untuk *monitoring* dan manajemen *job* di PT. AAA. Berdasarkan *Use Case Diagram* pada gambar 3, terdapat serangkaian langkah atau tindakan yang dijelaskan secara detail untuk setiap use case yang ada. Pertama, *use case* "Login Sistem" menjelaskan proses login tiap pengguna dan juga *admin* ke dalam sistem Jenkins. Pengguna harus memiliki akun dan setelah memasukkan *username* dan *password*, sistem akan melakukan validasi dan mengarahkan pengguna ke *dashboard* jika berhasil. Selanjutnya, use case "Lihat Semua *Job*" menjelaskan langkah untuk melihat daftar semua *job* yang ada di sistem oleh pengguna IT. *Pre-condition*-nya adalah pengguna harus memiliki *role* "IT" dan sudah login. Kemudian, "*Run* Semua *Job*" menjelaskan cara menjalankan job oleh pengguna IT dengan *pre-condition* yang sama. *Use case* "*Create Job*" menjelaskan cara membuat *job* baru oleh pengguna IT. Setelah mengakses *dashboard*, pengguna mengklik tombol "*New Item*" dan mengisi detail *job* sebelum menyimpannya. "*Delete Job*" menjelaskan cara menghapus *job* yang sudah ada, sedangkan "*Update Job*" menjelaskan cara mengubah *pipeline script* dari sebuah *job*. Selanjutnya, "Lihat *History Error* Semua *Job*" menjelaskan cara melihat riwayat *error* dari semua *job* oleh pengguna IT. "Lihat *History Error Job* Sesuai Divisi" menjelaskan cara yang sama namun khusus untuk pengguna non-IT, sedangkan "Lihat *Job* Sesuai Divisi" menjelaskan cara melihat daftar *job* sesuai dengan divisi pengguna non-IT. Terakhir, "*Run Job* Sesuai Divisi" menjelaskan cara menjalankan *job* sesuai dengan divisi pengguna non-IT. Semua *use case* ini membantu dalam manajemen *job* dan pengembangan perangkat lunak di sistem Jenkins.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.832




Gambar 4. Tampilan *Diagram Activity*

*Diagram Activity* pada gambar 4 menjelaskan implementasi *continuous integration* dan otomatisasi *monitoring job server* dengan Jenkins dimulai dengan *login* pengguna. Jenkins melakukan verifikasi kredensial dan peran pengguna. Jika kredensial tidak valid, pengguna akan diarahkan kembali ke halaman *login*. Jika kredensial valid, Jenkins akan memeriksa *role* pengguna dan menampilkan *dashboard* sesuai perannya. Ketika pengguna memilih layanan atau *job*, Jenkins akan mengirimkan *script* ke *server* untuk dieksekusi. *Server* akan memberikan respons dengan nilai 0 atau 1 (sukses atau gagal). Jika eksekusi gagal, *server* akan memberi tahu Jenkins, yang kemudian akan mengirimkan *email* otomatis kepada pengguna IT untuk melakukan pemeliharaan. Jika eksekusi berhasil, *server* akan memberikan respons berhasil kepada pengguna. Dari hal tersebut pengguna dapat memantau hasil dari proses CI dan otomatisasi *monitoring job server* yang telah dilakukan di tampilan *Stage View Job* yang dipilih.

Pada tahapan implementasi ini, *Compute Engine* digunakan sebagai *server* utama untuk menjalankan Jenkins untuk memantau dan mengelola proses *job* pada *server-server* PT. AAA yang terdistribusi. Tahapan selanjutnya merupakan konfigurasi *server Compute Engine* meliputi penyesuaian *setting* koneksi serta konfigurasi secara keseluruhan pada *Compute Engine*. Langkah ini penting untuk memastikan bahwa *server* dapat berfungsi secara optimal dalam mendukung operasional Jenkins dan proses *job* yang berjalan di perusahaan, sehingga memungkinkan pengelolaan infrastruktur *on-premises* yang lebih efisien dan efektif. Selain itu, beberapa *plugin* penting seperti *Pipeline Utility Steps*, *Email Extension Plugin*,

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.832

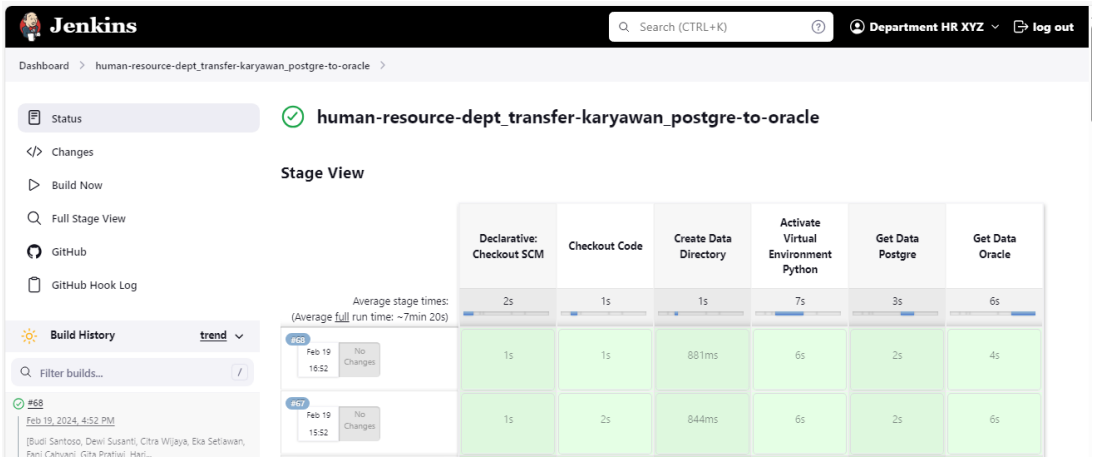
*PostgreSQL Database Plugin*, dan *Role-based Authorization Strategy* di *install* pada *server* Compute Engine. Langkah ini memastikan bahwa Jenkins dapat berfungsi secara optimal dalam memantau dan mengelola proses *job* dengan efisien. Tidak hanya itu, beberapa *library* Python seperti *psycopg2* dan *cx-Oracle* juga di *install* untuk mendukung proses *job* yang berjalan dan integrasi antar infrastruktur. Selanjutnya, file Jenkins file dibuat untuk digunakan oleh Jenkins dalam menjalankan proses *job* yang dikenal sebagai *Pipeline Jobs*. *Pipeline Jobs* ini berisi beberapa *stage* penting seperti *checkout SCM*, *preparing & installing environment*, dan juga *stage* menjalankan *script Python* seperti pada Kode Program 1 dibawah ini.

Kode Program 1. *Pipeline* Jenkins file

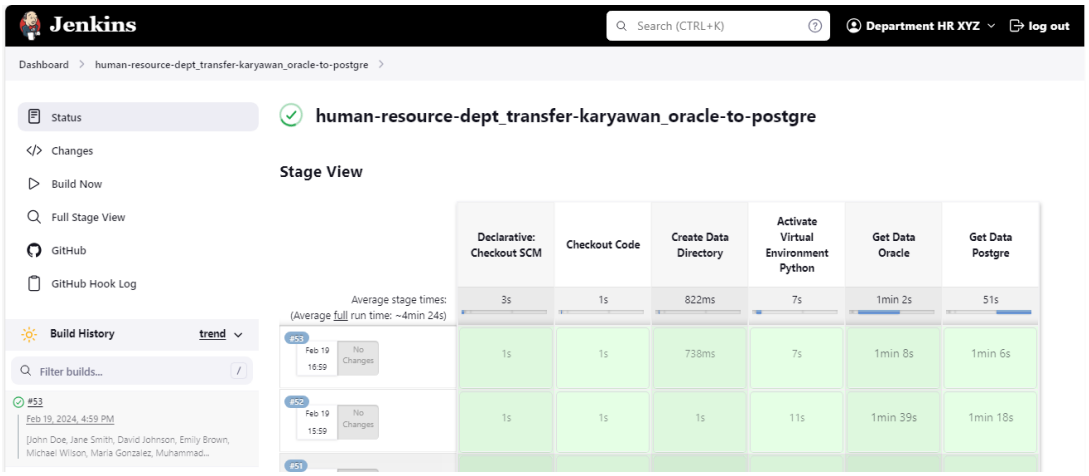
```
stages {
  stage('Checkout Code') {
     steps {
        checkout scm
     }
  }
  stage('Activate Virtual Environment Python') {
     steps {
         script {
             def workspaceBin = "${WORKSPACE}/venv/bin"
             env.PATH = "${workspaceBin}:${env.PATH}"
             sh "python3 -m venv ${WORKSPACE}/venv"
             sh ". ${WORKSPACE}/venv/bin/activate"
          }
     }
  }
  stage('Jalankan Python') {
     steps {
         script {
            def scriptPath ="${WORKSPACE}/selectOra.py"
            def output = sh(script:"${WORKSPACE}/path/python ${scriptPath}",
returnStdout: true).trim()
            echo "Python Output:\n${output}"
         }
     }
  }
}
```

Pada *stage* awal merupakan bagian untuk mengambil kode sumber dari *repository Source Code Management* (SCM) yang terhubung dengan *repository* git yang sudah disediakan dan dikirim ke dalam lingkungan kerja yang digunakan untuk melakukan *build, test,* dan *deployment* oleh Jenkins. Ini diperlukan agar Jenkins memiliki versi terbaru dari kode sumber untuk diproses. Hal tersebut memungkinkan Jenkins untuk menjalankan tugas-tugas yang telah didefinisikan tanpa adanya intervensi manual. Lalu *stage* berikutnya berisi perintah untuk mengaktifkan dan tempat untuk melakukan instalasi *library* Python dan memastikan kebutuhan yang diperlukan diperlukan untuk menjalankan proses integrasi antara sistem *on-premises* dan *cloud* telah terinstal dengan baik. Hal ini penting untuk memastikan bahwa proses integrasi dapat berjalan lancar dan efisien. Kemudian, *stage* menjalankan *script python*, di mana *script* tersebut digunakan untuk melakukan transaksi data antara *database* sistem *on-premises* dan *cloud*, atau untuk menjalankan tugas-tugas lain yang diperlukan dalam integrasi tersebut.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.832




Gambar 5. Tampilan *Stage View Job* Postgre ke Oracle




Gambar 6. Tampilan *Stage View Job* Oracle ke Postgre

Dari hal tersebut terlihat pada Gambar 5 dan Gambar 6 menunjukkan bahwa proses *job* berhasil dieksekusi untuk melakukan transaksi data sebanyak 1.000 data dari *database* PostgreSQL ke *database* Oracle. Begitu juga, sebaliknya, ditunjukkan pada Gambar 6 bahwa proses *job* berhasil dieksekusi untuk melakukan transaksi data sebanyak 20.000 data dari *database* Oracle ke *database* PostgreSQL. Selain itu, tampilan *stage view* ini juga menunjukkan durasi waktu setiap tahapan yang dieksekusi dan juga *history build* sebelumnya sehingga memudahkan untuk melacak kinerja dan histori dari setiap *job* yang dieksekusi sekarang maupun sebelumnya. Dengan informasi yang tersedia pada *stage view*, tim pengembang dapat mengevaluasi efisiensi proses, mengidentifikasi potensi perbaikan, dan memastikan bahwa proses *job* berjalan sesuai dengan yang diharapkan. Selain itu, informasi durasi waktu setiap tahapan juga dapat digunakan untuk melakukan analisis kinerja dan perencanaan kapasitas untuk meningkatkan efisiensi proses *job* di masa mendatang.

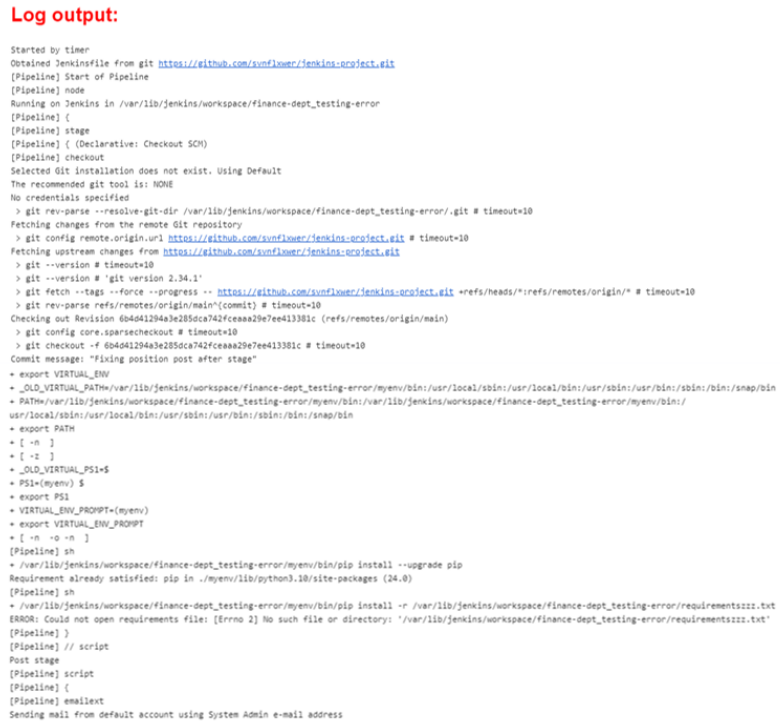
Kode Program 2. *Pipeline Post Failure* Jenkins file

```
post {
   failure {
       script {
         withCredentials([
            usernamePassword(
               credentialsId: 'gmail',
               usernameVariable: 'SMTP_USERNAME',
               passwordVariable: 'SMTP_PASSWORD'
```

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.832

```
            )
         ])
         {emailext (
            subject: """
               Build Failed: ${currentBuild.fullDisplayName}
(${env.BUILD_NUMBER}) """,
            body: """
               <html>
                  <body>
                    <h1 style="color:red"> Log: </h1>
                      <p><pre>\${BUILD_LOG}</pre></p>
                  </body>
               </html>
            """,
            to: "email@domain",
            replyTo: "email@domain",
            mimeType: 'text/html'
         )}
      }
   }
}
```

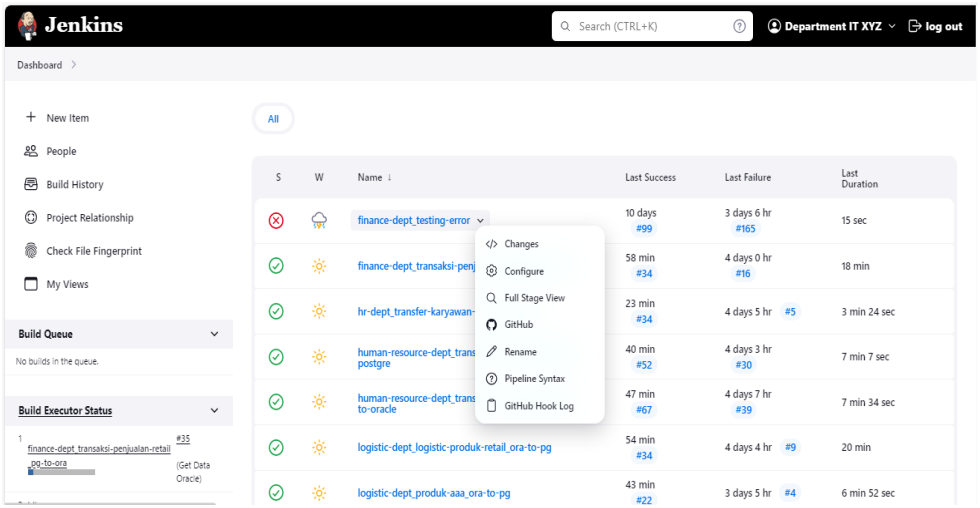
Selain itu, implementasi *job monitoring* ini memanfaatkan *post-failure* dan *plugin Email Extension Plugin* di setiap *stage* yang berjalan seperti pada Kode Program 2 diatas untuk memantau proses *job* yang berjalan pada *server-server* terdistribusi. Hal ini memungkinkan PT. AAA untuk secara proaktif mendeteksi dan menangani masalah yang mungkin timbul dalam proses job mereka. Jenkins dikonfigurasi untuk mengirimkan *email* otomatis kepada PIC yang terkait jika terjadi masalah dengan proses *job*, sehingga memungkinkan respons yang cepat dan efektif. Hasil dari implementasi ini telah terlihat pada Gambar 7. Ketika terjadi kesalahan dalam proses *job*, *Jenkins* akan secara otomatis mengirimkan *email* notifikasi kepada PIC yang terkait. *Email* notifikasi ini berisi *log error* yang memungkinkan tim IT PT. AAA untuk segera menanggapi dan menangani masalah yang terjadi.



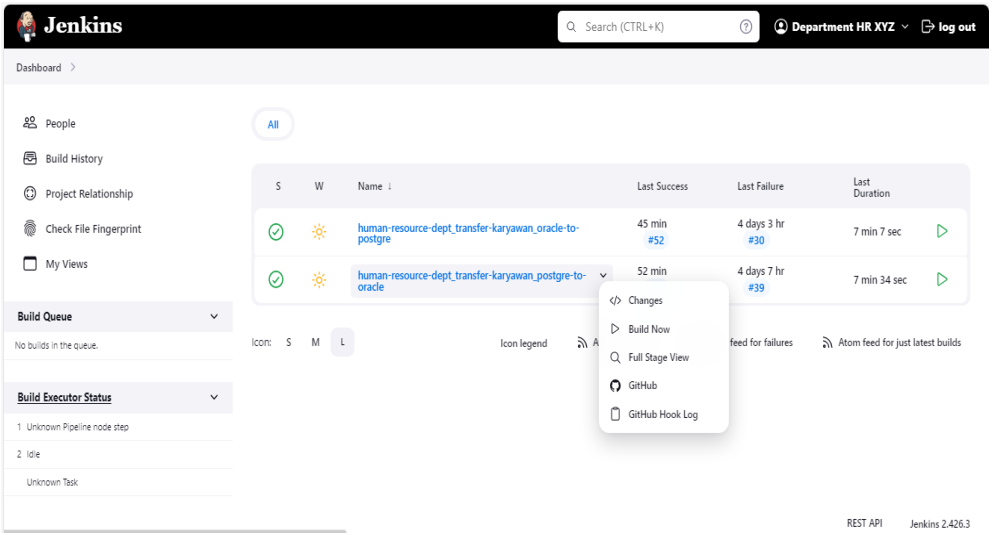

Gambar 7. *Email Log Output Error*

**Jurnal Indonesia : Manajemen Informatika dan Komunikasi**
https://journal.stmiki.ac.id
E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.832

Untuk mengelola akses, dibuat *privilege* akses *user* yang sesuai untuk mengakses dan mengelola Jenkins serta proses *job* yang berjalan. Hak akses ini telah diatur oleh *user* admin yang berperan sebagai *administrator* sistem. *User IT* memiliki kemampuan untuk melihat semua *job*, menjalankan semua *job*, membuat *job* baru, menghapus *job*, memperbarui *job*, dan melihat riwayat *error* dari semua *job* yang ada. Di sisi lain, *user Non-IT* hanya dapat melihat riwayat *job* yang *error* sesuai dengan divisi mereka, melihat *job* sesuai dengan divisi, dan menjalankan *job* sesuai dengan divisi mereka yang terlihat seperti pada Gambar 8 dan gambar 9 dibawah.




Gambar 8. Tampilan *Dashboard User IT*




Gambar 9. Tampilan *Dashboard User Non-IT*

Perlu dilakukan tahap pengujian sebelum masuk ke tahap produksi agar memastikan perangkat lunak berjalan dengan baik. Hasil dari pengujian sistem yang dilakukan menggunakan metode *blackbox testing* oleh tim *Quality Assurance* PT. AAA dapat dilihat dari tabel 1 dibawah ini.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.832

Tabel 1. Pengujian *Blackbox*

| Test Skenario | Hasil yang diharapkan | Status Pengujian |
|---|---|---|
| Menjalankan Jenkins *job* secara manual | Jenkins dapat menjalankan *job* secara manual dan menghasilkan *output* yang sesuai dengan langkah-langkah yang dijalankan | *Pass* |
| Mengirimkan *email* notifikasi ketika *job* gagal dieksekusi | Jenkins dapat mengirimkan *email* notifikasi kepada PIC yang terkait ketika terjadi kesalahan dalam eksekusi *job* | *Pass* |
| Memantau proses *job* secara *real-time* | Jenkins dapat memantau proses *job* secara *real-time* dan menampilkan status yang akurat sesuai dengan tahapan yang sedang berjalan | *Pass* |
| Mengaktifkan integrasi antara sistem *on-premises* dan *cloud* | Jenkins dapat melakukan integrasi antara sistem *on-premises* dan *cloud* dengan mengambil kode sumber dari *repository* git dan menjalankan proses *job* yang sesuai dan menyiapkan *requirement* tiap sistem. | *Pass* |
| Menjalankan *script python* untuk transaksi data antara *database* sistem *on-premises* dan *cloud* | Jenkins dapat menjalankan *script* Python untuk mentransfer data antara *database* sistem *on-premises* dan *cloud* dengan benar | *Pass* |
| Mengelola akses *user* secara tepat | Jenkins dapat mengelola akses *user* sesuai dengan peran masing-masing, termasuk *user IT* dan *Non-IT* | *Pass* |

Dari hasil tabel 1 Pengujian *Blackbox*, setiap tes skenario yang dilakukan mencakup satu fungsi atau fitur yang diuji untuk memastikan bahwa implementasi Jenkins dan integrasi dengan infrastruktur *on-premises* dan *cloud* berfungsi sesuai dengan yang diharapkan. Setiap tes skenario memiliki deskripsi yang jelas tentang langkah-langkah yang diambil dan hasil yang diharapkan. Setelah tes dilakukan, hasilnya akan dicatat sebagai *'Pass'* jika fungsi atau fitur berjalan sesuai dengan yang diharapkan, atau *'Fail'* jika terjadi masalah dalam eksekusi. Berdasarkan hasil pengujian yang dilakukan, semua uji coba yang dilakukan berhasil dengan tingkat keberhasilan 100%, menunjukkan bahwa perangkat lunak berjalan dengan baik dan memenuhi kebutuhan awal yang ditetapkan.

## 4. Kesimpulan

Berdasarkan hasil penelitian, penerapan Jenkins dalam pemantauan dan otomatisasi proses job sinkronisasi antara *database on-premises* dan *cloud* telah berhasil dilakukan oleh PT. AAA. Dengan menggunakan Compute Engine sebagai *server* utama, konfigurasi *server* yang optimal, instalasi *plugin* dan *library* yang diperlukan, serta pembuatan jenkinsfile, PT. AAA berhasil memantau dan mengelola proses job dengan efisien. Proses integrasi antara sistem *on-premises* dan *cloud* juga berjalan lancar, terlihat dari keberhasilan transaksi data antara *database* PostgreSQL dan Oracle. Penggunaan *plugin Email Extension Plugin* untuk notifikasi otomatis dan manajemen akses pengguna dengan *Role-based Authorization Strategy* juga telah membantu PT. AAA dalam mengatasi masalah yang mungkin timbul dan meningkatkan efektivitas operasional. Dengan demikian, penerapan Jenkins dalam pemantauan

E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.832

dan otomatisasi proses job sinkronisasi antara *database on-premises* dan *cloud* telah sesuai dengan kebutuhan awal yang ditetapkan oleh PT. AAA. Hasil penelitian ini memberikan gambaran bahwa PT. AAA dapat mengoptimalkan *monitoring* dan manajemen proses *job* secara efisien dan efektif, serta siap untuk melangkah ke tahap produksi.

## 5. Ucapan Terima Kasih

Ucapan terima kasih penulis sampaikan kepada pemberi dana penelitian dan donatur yang telah memberikan dukungan finansial. Penulis juga ingin mengucapkan terima kasih kepada semua pihak yang telah membantu dalam pelaksanaan penelitian ini.

## 6. Daftar Pustaka

Ahuja, S. P., Czarnecki, E., & Willison, S. (2020). Multi-factor performance comparison of amazon web services elastic compute cluster and google cloud platform compute engine. *International Journal of Cloud Applications and Computing (IJCAC)*, *10*(3), 1-16.

Antunes, R. V., Navarro, G. M., & Hanazumi, S. (2018, September). Test Framework for Jenkins Shared Libraries. In *Proceedings of the III Brazilian Symposium on Systematic and Automated Software Testing* (pp. 13-19). DOI: https://doi.org/10.1145/3266003.3266008.

Armenise, V. (2015, May). Continuous delivery with Jenkins: Jenkins solutions to implement continuous delivery. In *2015 IEEE/ACM 3rd International Workshop on Release Engineering* (pp. 24-27). IEEE. DOI: http://dx.doi.org/10.1109/RELENG.2015.19.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, *21*(5), 61-72. DOI: http://dx.doi.org/10.1109/2.59.

Cloud, S. Q. L. (2018). Google Cloud Platform.

Eessaar, E. (2021). The usage of declarative integrity constraints in the SQL databases of some existing software. In *Software Engineering and Algorithms: Proceedings of 10th Computer Science On-line Conference 2021, Vol. 1* (pp. 375-390). Springer International Publishing.

Fisher, C. (2018). Cloud versus on-premise computing. *American Journal of Industrial and Business Management*, *8*(9), 1991-2006.

Gupta, B., Mittal, P., & Mufti, T. (2021, March). A review on amazon web service (aws), microsoft azure & google cloud platform (gcp) services. In *Proceedings of the 2nd International Conference on ICT for Digital, Smart, and Sustainable Development, ICIDSSD 2020, 27-28 February 2020, Jamia Hamdard, New Delhi, India*. DOI: http://dx.doi.org/10.4108/eai.27-2-2020.2303255.

Narayani, N., Kumar, P., & Kumar, D. (2022, December). Web Scraping & Automation Bot Using Python: Using Python to automate all the tasks. In *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)* (pp. 1343-1346). IEEE. DOI: https://doi.org/10.1109/ICAC3N56670.2022.10074375.

Pressman, R. S. (2005). *Software engineering: a practitioner's approach*. Palgrave macmillan.

E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.832

Pressman, R. S. (2012). Rekayasa perangkat lunak: pendekatan praktisi.

Royce, W. W. (1970, August). Managing the development of large systems: Concepts and techniques. In *9th International Conference on Software Engineering. ACM* (pp. 328-38).

Saragih, R. R. (2016). Pemrograman dan bahasa Pemrograman. *STMIK-STIE Mikroskil*, 1-91.

Sugiyono, S. (2013). Metode penelitian kualitatif. *Bandung: Alfabeta*.