PERANCANGAN *CLOUD BACKUP* APLIKASI KEPUASAN PASIEN BERBASIS DOCKER DENGAN *GOOGLE CLOUD PLATFROM*

Muhammad Izzudin Farhans [1], Andre Letto [2], Agussalim [3], Anggraini Puspita Sari [4], Ikrimatul Ulumiyyah [5]
[1,2,3,4] Magister Teknologi Informasi, Universitas Pembangunan Nasional "Veteran" Jawa Timur,
[5] Bagian Administrasi Pembangunan, Sekretariat Daerah Kabupaten Lamongan
24066020009@student.upnjatim.ac.id

ABSTRAK

Keamanan dan kontinuitas data merupakan aspek krusial dalam pengelolaan aplikasi layanan publik, terutama pada sistem kepuasan pasien yang menangani data sensitif. Penelitian ini membahas perancangan dan implementasi sistem cloud backup untuk aplikasi kepuasan pasien di Puskesmas dengan memanfaatkan teknologi Docker dan Google Cloud Platform. Sistem ini mengintegrasikan database MySQL dan aplikasi berbasis CodeIgniter dalam container Docker, serta mengotomatiskan proses pencadangan ke Google Drive menggunakan skrip Python yang dijalankan melalui Cron Job. Pengujian dilakukan dengan skema simulasi backup setiap lima menit selama 40 menit untuk mengevaluasi stabilitas, kecepatan, dan keandalan sistem. Hasil pengujian menunjukkan bahwa seluruh proses backup berjalan sukses dengan tingkat keberhasilan 100%, durasi proses berkisar antara 6,26 detik hingga 29,23 detik, dan pencatatan log yang lengkap. Meskipun sistem telah memenuhi tujuan pencadangan otomatis tanpa intervensi manual, ditemukan peluang perbaikan pada aspek efisiensi waktu dan manajemen file, seperti penerapan incremental backup, peningkatan keamanan kredensial, dan pengaturan arsip file lama. Sistem ini memberikan solusi portabel dan andal untuk mendukung keberlangsungan data aplikasi kepuasan pasien di lingkungan pelayanan kesehatan.

Keyword : *Cloud Backup, Docker, Google Cloud Platform, MySQL, Cron Job, Otomatisasi*

1. PENDAHULUAN

Kepuasan pasien merupakan indikator krusial dalam menilai kualitas layanan kesehatan, karena mencerminkan persepsi dan pengalaman pasien terhadap mutu pelayanan yang diterima. Seiring dengan berkembangnya teknologi informasi, berbagai institusi kesehatan mulai mengadopsi aplikasi digital untuk mengukur kepuasan pasien secara sistematis. Penggunaan aplikasi semacam ini memungkinkan pengumpulan dan analisis data secara real-time, sehingga mendukung proses evaluasi mutu layanan yang lebih cepat, akurat, dan berbasis data [1]**.** Namun demikian, data yang dikumpulkan oleh aplikasi kepuasan pasien umumnya mengandung informasi sensitif, seperti identitas pasien dan opini terhadap layanan medis yang diterima. Tanpa sistem perlindungan data yang memadai, informasi ini rentan terhadap risiko kehilangan, kerusakan, atau kebocoran, yang dapat berdampak serius terhadap kepercayaan publik dan reputasi institusi layanan kesehatan. Oleh karena itu, aspek keamanan dan keandalan sistem pencadangan data menjadi sangat penting dalam infrastruktur teknologi informasi rumah sakit dan puskesmas [2]**.**

Penerapan strategi backup yang efektif merupakan langkah strategis untuk menjamin ketersediaan dan integritas data layanan kesehatan. Solusi backup berbasis *cloud* saat ini banyak diadopsi karena menawarkan keunggulan dari sisi fleksibilitas, skalabilitas, dan efisiensi biaya [3]. Teknologi ini memungkinkan replikasi data secara otomatis, dukungan terhadap pemulihan bencana (disaster recovery), serta pemenuhan parameter teknis seperti *Recovery Point Objective* (RPO) dan *Recovery Time Objective* (RTO) sesuai kebutuhan sistem kritikal. Namun demikian, implementasi backup *cloud* juga menghadapi sejumlah tantangan, seperti konsistensi snapshot, proteksi terhadap data *corruption*, dan pemenuhan regulasi privasi data [4].

Dalam mendukung efisiensi pengembangan dan pengelolaan aplikasi modern, teknologi *containerization* seperti Docker telah menjadi pendekatan yang banyak digunakan. Docker memungkinkan aplikasi dan seluruh dependensinya dikemas ke dalam *container* yang ringan, terisolasi, dan portabel, sehingga dapat dijalankan secara konsisten di berbagai lingkungan, baik lokal maupun *cloud* [5]**.** Integrasi antara arsitektur *container* dengan platform *cloud* memberikan peluang besar untuk mengotomatisasi proses backup serta mengimplementasikan strategi pemulihan data yang lebih adaptif dan handal.

Google Cloud Platform (GCP) menyediakan berbagai layanan dan API yang mendukung pengembangan serta pengelolaan aplikasi berbasis container. Layanan seperti *Cloud Storage, Cloud Scheduler*, dan Google Drive API memungkinkan proses pencadangan dan pemulihan data berjalan otomatis dan aman [6]**.** Selain itu, fitur keamanan seperti enkripsi data, *Identity and Access Management* (IAM), serta monitoring dan logging membantu meningkatkan visibilitas serta kontrol terhadap seluruh proses *backup* [7]**.**

Dengan mempertimbangkan kebutuhan sistem aplikasi kepuasan pasien yang berjalan di atas arsitektur *container*, maka diperlukan perancangan

solusi *backup* yang terautomasi, aman, dan dapat diandalkan. Penelitian ini bertujuan untuk merancang dan mengimplementasikan sistem *cloud backup* untuk aplikasi kepuasan pasien berbasis Docker dengan memanfaatkan layanan Google *Cloud Platform*. Harapannya, solusi ini dapat memastikan kontinuitas layanan, melindungi data sensitif, serta meminimalkan risiko kehilangan data akibat kegagalan sistem atau insiden keamanan..

2. TINJAUAN PUSTAKA

2.1. Teknologi *Cloud Computing*

Layanan *cloud computing* dihadirkan sebagai upaya untuk memungkinkan akses sumber daya dan aplikasi dari mana saja melalui jaringan Internet di era digital. *Cloud computing* memungkinkan pengguna mengakses sumber daya komputasi seperti server, penyimpanan, dan aplikasi tanpa perlu memiliki infrastruktur fisik sendiri [8]. Layanan *cloud* dibagi menjadi tiga kategori utama: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), dan Software as a Service (SaaS), yang masing-masing menawarkan fleksibilitas dan efisiensi dalam penyediaan layanan TI.

Pemanfaatan *cloud computing* dalam sektor publik menjadi semakin relevan mengingat kebutuhan akan layanan digital yang cepat, efisien, dan terjangkau. Ekaputra dan Affandi menekankan bahwa penggunaan *cloud* dalam konteks layanan publik memberikan keunggulan dari sisi skalabilitas, namun tetap memiliki tantangan dalam aspek keamanan dan kontrol data [9].

2.2. *Google Cloud Platform*

Google Cloud Platform (GCP) adalah salah satu penyedia layanan *cloud* terkemuka yang menyediakan berbagai layanan seperti Google Drive API, *Identity and Access Management* (IAM), serta monitoring dan logging. GCP memfasilitasi integrasi layanan backup berbasis *cloud* dengan skala dan fleksibilitas tinggi.

Studi oleh adelia menjelaskan bahwa GCP menawarkan pendekatan serverless yang efektif dalam mendukung pengelolaan data dan backup otomatis [10]. Dibandingkan dengan penyedia lain seperti AWS dan Azure, GCP unggul dalam integrasi API dan harga kompetitif [11]

2.3. Teknologi *Container* dan Docker

Docker merupakan teknologi *container*ization yang memungkinkan aplikasi berjalan dalam lingkungan yang terisolasi dan ringan. Sholihah dan Darujati membuktikan efektivitas Docker dalam replikasi database MySQL secara efisien [12]. Penggunaan Docker dalam pengembangan aplikasi meningkatkan konsistensi lingkungan serta mempercepat proses deployment

2.4. Manajemen dan Automasi Backup

Chandra menyoroti bahwa penerapan automasi backup melalui penggunaan script yang dijalankan secara terjadwal dengan cron serta disertai mekanisme pencatatan (logging), dapat secara signifikan mengurangi risiko kehilangan data akibat kesalahan manusia (human error), khususnya dalam pengelolaan perangkat pada sistem jaringan *Internet Service Provider* (ISP) [13].

2.5. Integrasi *Cloud* Backup dengan Aplikasi Kesehatan

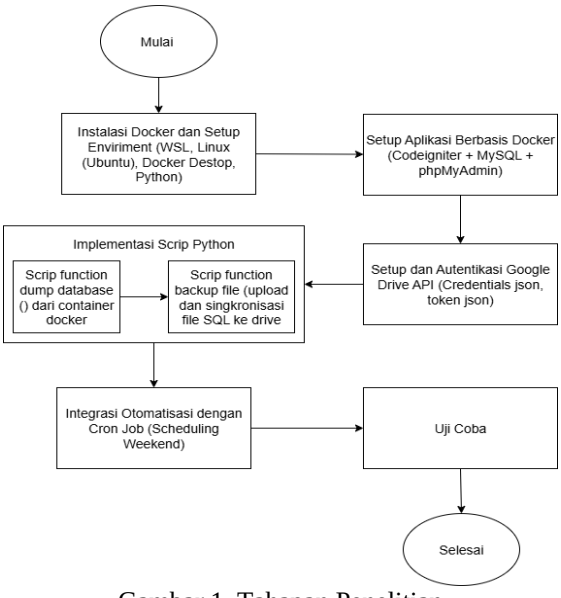
Implementasi sistem e-health berbasis *cloud* tidak terlepas dari berbagai tantangan keamanan informasi yang signifikan. Di antara tantangan tersebut adalah potensi kehilangan data, kebocoran informasi sensitif, serta risiko pembajakan akun pengguna oleh pihak yang tidak berwenang. Kondisi ini menunjukkan bahwa keberhasilan penerapan teknologi *cloud* dalam layanan kesehatan sangat bergantung pada penerapan sistem keamanan yang komprehensif [14]**.**

3. METODE PENELITIAN

3.1. Jenis Penelitian

Penelitian ini menerapkan metode eksperimen studi kasus untuk menguji otomatisasi proses pencadangan (backup) data kepuasan pasien dari aplikasi kepuasan pasien pelayanan Puskesmas. Sistem pencadangan ini melibatkan database MySQL yang berjalan dalam *container* Docker bersamaan dengan aplikasi tersebut, dan diintegrasikan dengan penyimpanan *cloud* Google Drive melalui script Python.

3.2. Tahapan Penelitian



Gambar 1. Tahapan Penelitian

Pada tahapan ini merupakan proses pengembangan dan integrasi aplikasi berbasis Docker yang terhubung dengan Google Drive API. Diagram ini memvisualisasikan langkah-langkah mulai dari instalasi lingkungan kerja, pengaturan aplikasi, hingga implementasi otomatisasi menggunakan *Cron Job*, serta uji coba akhir sebelum sistem dinyatakan selesai.

3.3. Instalasi Docker dan Setup Environment.

Tahap awal dimulai dengan melakukan instalasi dan konfigurasi environment yang mendukung proses pengembangan dan otomatisasi. Lingkungan yang digunakan adalah sistem operasi Windows dengan dukungan WSL (Windows Subsystem for Linux) berbasis Ubuntu, Docker Desktop sebagai platform kontainerisasi, dan Python sebagai bahasa pemrograman yang digunakan untuk penjadwalan dan eksekusi otomatisasi.

3.4. Setup Aplikasi Berbasis Docker

Setelah lingkungan siap, selanjutnya dilakukan pembuatan aplikasi berbasis Docker. Aplikasi yang digunakan terdiri dari:

a. CodeIgniter sebagai framework web berbasis PHP.
b. MySQL sebagai sistem manajemen basis data.
c. phpMyAdmin sebagai antarmuka grafis untuk pengelolaan database.

Seluruh komponen tersebut dijalankan dalam container Docker yang saling terhubung, sehingga memudahkan pengelolaan dan isolasi lingkungan aplikasi.

3.5. Setup dan Autentikasi Google Drive API

Untuk menyimpan hasil backup di cloud, Google Drive dipilih sebagai media penyimpanan. Proses ini memerlukan pembuatan credentials di Google API Console dan menghasilkan file credentials.json serta token.json sebagai otorisasi akses. File ini memungkinkan script Python untuk terhubung ke akun Google Drive pengguna secara otomatis

3.6. Implementasi Script Python

Tahap ini merupakan bagian inti dari sistem. Dua fungsi utama diimplementasikan menggunakan Python:

- Dump Database: Script mengambil cadangan data dari database MySQL di dalam container Docker.
- Backup ke Google Drive: Script selanjutnya akan mengunggah file hasil dump tersebut ke Google Drive, serta melakukan sinkronisasi jika terjadi perubahan.

3.7. Integrasi Otomatisasi dengan Cron Job

Agar proses backup berjalan secara terjadwal, dilakukan integrasi dengan Cron Job. Jadwal yang ditentukan adalah setiap akhir pekan (weekend), untuk memastikan data terkini selalu dicadangkan secara berkala. Cron Job akan memanggil script Python sesuai waktu yang telah diatur

3.8. Uji Coba

Untuk memastikan bahwa sistem backup otomatis berjalan sesuai harapan, dilakukan uji coba dengan skema siklus cronjob yang dipercepat. Dalam uji coba ini, waktu eksekusi cronjob diatur setiap 5 menit sekali selama rentang waktu 40 menit. Tujuannya adalah untuk mensimulasikan siklus backup mingguan dalam waktu yang lebih singkat guna mengobservasi perilaku sistem. mengevaluasi keandalan skrip backup, serta memastikan integrasi dengan Google Drive berjalan dengan baik.

Skrip backup.py dimodifikasi untuk mencatat waktu mulai dan selesai proses backup, nama file backup yang dihasilkan, serta informasi hasil upload ke Google Drive ke dalam berkas log (backup.log). Selain itu, penamaan file backup selama uji coba menggunakan prefix test_ agar hasil uji dapat dibedakan dari backup reguler.

Cronjob diatur menggunakan sintaks */5 * * * *, yang mengeksekusi perintah setiap lima menit. Selama 40 menit durasi uji coba, diharapkan terdapat setidaknya delapan entri log hasil backup, yang masing-masing memuat bukti bahwa proses dump database dan upload ke cloud berhasil dilakukan tanpa intervensi manual.

Hasil dari uji coba ini digunakan untuk mengevaluasi stabilitas, ketepatan jadwal, dan keandalan sistem pencadangan otomatis sebelum diimplementasikan dalam siklus mingguan yang sebenarnya (0 3 * * 0).

3.9. Tools yang digunakan

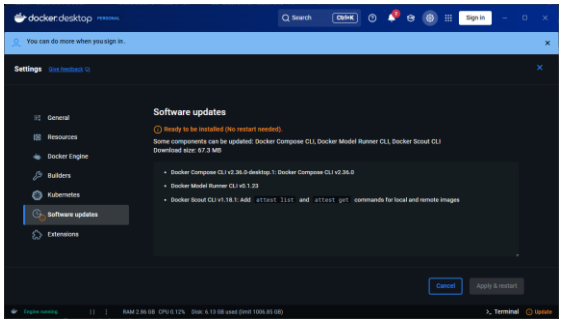
Untuk mendukung proses perancangan dan implementasi sistem *cloud* backup pada aplikasi kepuasan pasien, berbagai tools dan teknologi dipilih secara strategis berdasarkan kebutuhan sistem yang mengedepankan otomatisasi, portabilitas, dan keamanan data. Pemilihan tools ini bertujuan untuk memastikan integrasi yang efisien antara aplikasi berbasis Docker dan layanan *cloud* dari Google, serta mendukung proses backup yang berjalan secara otomatis tanpa mengganggu kinerja layanan utama. Berikut adalah daftar tools yang digunakan dalam sistem yang dibangun

a. WSL
b. Linux Ubuntu
c. Docker Desktop
d. Python Mysql
e. PhpMyAdmin
f. google drive api (google *cloud*)
g. cronjob (untuk automasi)

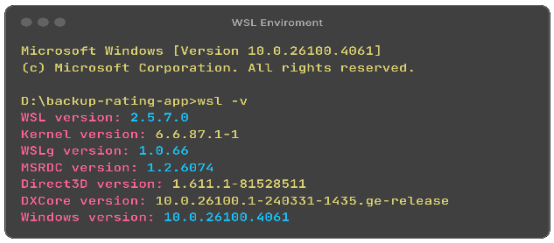
4. HASIL DAN PEMBAHASAN

4.1. Instalasi Docker dan Setup Enviroment (WSL, Linux (ubuntu), Docker Desktop, Python)

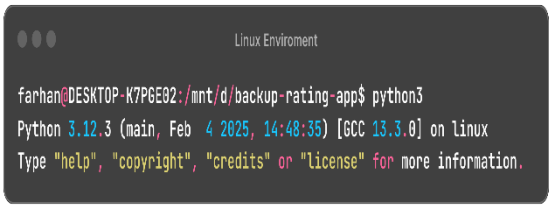
Langkah awal dilakukan dengan menyiapkan lingkungan kerja berbasis Linux melalui Windows Subsystem for Linux (WSL) dan distribusi Ubuntu. Docker Desktop digunakan untuk mengelola *container*, sedangkan Python dipasang sebagai alat bantu scripting automasi backup. Persiapan ini menjadi fondasi utama dalam memastikan kompatibilitas dan stabilitas system. Gambar di bawah ini menggambarkan tahapan instalasi yang diperlukan sebagai bagian dari proses konfigurasi awal lingkungan kerja berbasis Docker, WSL (Windows Subsystem for Linux), sistem operasi Linux Ubuntu, serta Python sebagai bahasa pemrograman utama dalam ekosistem tersebut.



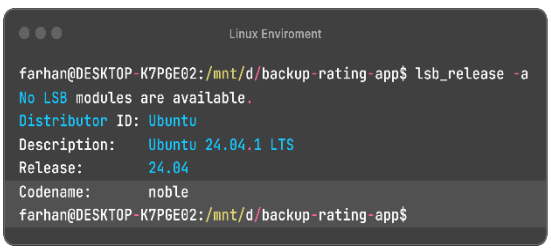
Gambar 2. Instalasi Docker



Gambar 3. WSL Enviroment



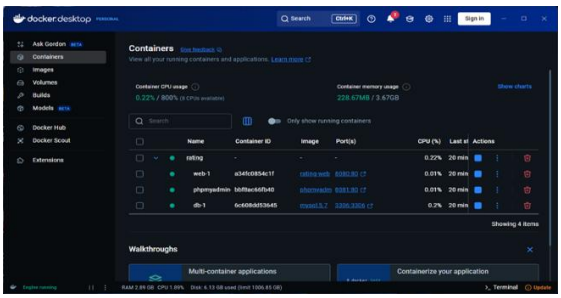
Gambar 4. Python



Gambar 5. Ubuntu

4.2. Setup Aplikasi Berbasis Docker (CodeIgniter + MySQL + phpMyAdmin)

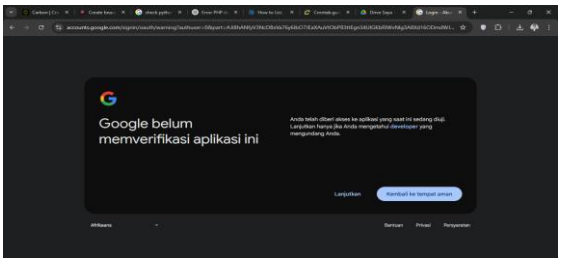
Aplikasi kepuasan pasien dibangun menggunakan framework CodeIgniter dan dikombinasikan dengan MySQL sebagai sistem basis data serta phpMyAdmin untuk pengelolaan database. Seluruh komponen dikemas dalam *container* menggunakan Docker Compose untuk memudahkan orkestrasi, isolasi, dan portabilitas antar lingkungan pengujian dan produksi. Gambar di bawah ini menunjukkan bahwa beberapa aplikasi berbasis kontainer telah berhasil dijalankan atau dikonfigurasi di dalam sistem Docker, yang ditandai dengan munculnya daftar kontainer aktif beserta informasi terkait seperti nama kontainer, image yang digunakan, port yang dipetakan, serta penggunaan sumber daya CPU dan memori.



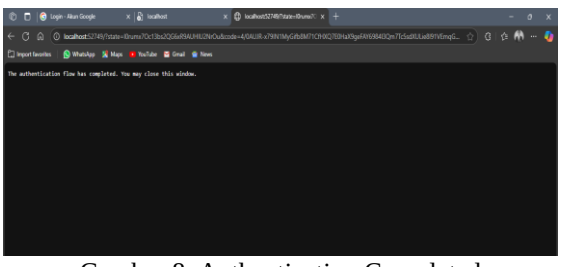
Gambar 6. Aplikasi Berjalan pada Docker

4.3. Setup dan Autentikasi Google Drive API (credentials.json, token.json)

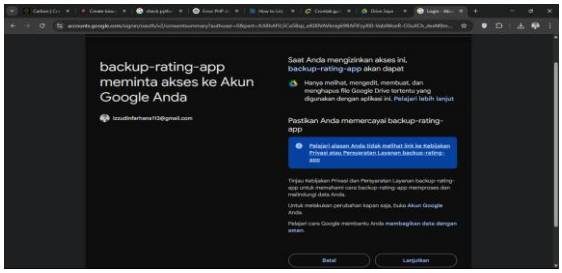
Proses autentikasi dan integrasi dengan Google Drive dilakukan melalui pemanfaatan Google Drive API. File credentials.json digunakan sebagai identitas aplikasi untuk mengakses layanan *cloud*, dan proses autentikasi menghasilkan token.json sebagai bentuk otorisasi yang aman. Folder tujuan pencadangan di Google Drive disiapkan dan diverifikasi di function login_google_drive.



Gambar 7. Verifikasi aplikasi ke google



Gambar 8. Authentication Complated



Gambar 9. Meminta akses ke google

4.4. Script function dump_database() dari *Container* Docker.

Skrip modul ini menjalankan perintah untuk menyimpan file backup. pada modul ini memanfaatkan beberapa library, yaitu datetime python untuk mendapatkan waktu sekarang dan memformat dalam bentuk string yang rapi.

Library os dimanfaatkan untuk medapatkan ukuran file hasil back up. Kemudian Library subprocess modul standar Python untuk menjalankan perintah sistem (command-line) dari dalam Python. Digunakan untuk menjalankan perintah docker exec mysqldump supaya data MySQL di dalam container Docker bisa diekspor.

```
def dump_database():
    now = datetime.now()
    timestamp = now.strftime('%Y-%m-%d_%H%M')
    output_file = f"{FOLDER_TO_BACKUP}/test_backup_{timestamp}.sql"

    cmd = [
        'docker', 'exec', CONTAINER_NAME,
        'mysqldump', '-u', DB_USER, f"-p{DB_PASSWORD}", DB_NAME
    ]
    with open(output_file, 'w') as f:
        subprocess.run(cmd, stdout=f)

    size_kb = os.path.getsize(output_file) / 1024
    size_str = f"{size_kb:.2f} KB" if size_kb < 1024 else f"{size_kb / 1024:.2f} MB"
    print(f"[{now.strftime('%Y-%m-%d %H:%M:%S')}] ✅ Database berhasil diexport ke:
{output_file} ({size_str})")
```

Gambar 10. Dump (ekspor) database MySQL dari dalam sebuah container Docker ke dalam file sql

4.5. Script function backup files (Upload & Sinkronisasi File SQL ke Google Drive)

```
def backup_files():
    service = login_google_drive()
    folder_id = '1yryPaHH9YBYytOYN8Lioy_ycqS3k5itg'
    print(f"[{datetime.now().strftime('%Y-%m-%d %H:%M:%S')}] 🚀 Memulai upload file ke
Google Drive...")

    for filename in os.listdir(FOLDER_TO_BACKUP):
        filepath = os.path.join(FOLDER_TO_BACKUP, filename)
        if os.path.isfile(filepath):
            size_kb = os.path.getsize(filepath) / 1024
            size_str = f"{size_kb:.2f} KB" if size_kb < 1024 else f"{size_kb / 1024:.2f}
MB"

            query = f"name = '{filename}' and '{folder_id}' in parents and trashed =
false"
            results = service.files().list(q=query, spaces='drive', fields="files(id,
name)").execute()
            items = results.get('files', [])
            media = MediaFileUpload(filepath, resumable=True)
            now = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

            if items:
                file_id = items[0]['id']
                service.files().update(fileId=file_id, media_body=media).execute()
                print(f"[{now}] 🔄 File diperbarui: {filename} ({size_str}) (ID:
{file_id})")
            else:
                file_metadata = {'name': filename, 'parents': [folder_id]}
                uploaded = service.files().create(body=file_metadata, media_body=media,
fields='id').execute()
                print(f"[{now}] ✅ File diupload: {filename} ({size_str}) (ID:
{uploaded.get('id')})")


if __name__ == '__main__':
    start = datetime.now()
    print(f"\n===== 🕒 Mulai backup pada {start.strftime('%Y-%m-%d %H:%M:%S')} =====")
    dump_database()
    backup_files()
    end = datetime.now()
    duration = end - start
    seconds = duration.total_seconds()
    minutes = seconds / 60

    durasi_str = f"{seconds:.2f} detik" if seconds < 60 else f"{minutes:.2f} menit"
    print(f"===== ✅ Backup selesai pada {end.strftime('%Y-%m-%d %H:%M:%S')} (dalam
{durasi_str}) =====\n")
```

Gambar 11. Sinkronisasi (upload atau update) file dari lokal ke Google Drive secara otomatis menggunakan API Google Drive

Skrip melakukan upload file hasil dump ke Google Drive menggunakan API, dengan opsi sinkronisasi atau overwrite file yang telah ada sebelumnya. Gambar 11 Proses modul backup_files()dimulai dengan melakukan login ke Google Drive melalui fungsi login_google_drive() di Gambar 12 untuk mendapatkan objek service yang digunakan berinteraksi dengan Google Drive API. Setelah itu, ditentukan folder_id sebagai lokasi tujuan penyimpanan di Google Drive, lalu sistem menampilkan pesan bahwa proses upload telah dimulai. Program kemudian membaca seluruh isi folder FOLDER_TO_BACKUP menggunakan os.listdir() dan memeriksa setiap item dengan os.path.isfile() agar hanya memproses file.

```
login google drive
def login_google_drive():
    creds = None
    if os.path.exists(TOKEN_FILE):
        creds = Credentials.from_authorized_user_file(TOKEN_FILE, SCOPES)
    if not creds or not creds.valid:
        flow = InstalledAppFlow.from_client_secrets_file(CREDENTIALS_FILE, SCOPES)
        creds = flow.run_local_server(port=0)
        with open(TOKEN_FILE, 'w') as token:
            token.write(creds.to_json())
    return build('drive', 'v3', credentials=creds)
```

Gambar 12. Autentikasi login ke Google Drive API

Setiap file yang ditemukan akan dihitung ukurannya menggunakan os.path.getsize() dan diformat menjadi KB atau MB. Selanjutnya, program membuat query ke Google Drive API untuk memeriksa apakah file dengan nama yang sama sudah ada di folder tujuan. Jika file ditemukan, maka program akan memperbaruinya menggunakan metode service.files().update(). Jika tidak ditemukan, maka file akan diunggah baru dengan service.files().create() beserta metadata folder tujuannya.

Selama proses, setiap aksi upload atau update akan ditampilkan di konsol lengkap dengan waktu, ukuran file, dan ID file di Google Drive. Pada bagian akhir, jika modul ini dijalankan langsung, proses dimulai dengan mencatat waktu mulai, memanggil dump_database() untuk membuat backup database, lalu memanggil backup_files() untuk mengunggah hasil backup. Setelah itu waktu selesai dicatat, durasi total proses dihitung, dan hasil ringkasan proses backup ditampilkan.

4.6. Integrasi Otomatisasi dengan Cron Job (Schedulig Weekend)

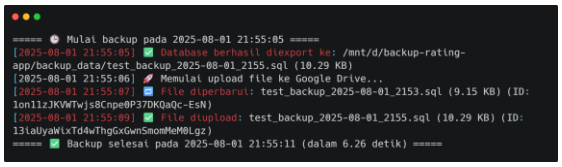
Untuk memastikan proses backup berjalan secara berkala tanpa intervensi manual, digunakan Cron Job pada sistem operasi Linux. Penjadwalan ditetapkan secara mingguan, khususnya pada akhir pekan, agar tidak mengganggu performa sistem saat jam operasional layanan. Pada Cron job dibawah ini digunakan untuk menjalankan skrip backup Python setiap Minggu jam 03:00 pagi, dan mencatat seluruh

hasilnya (baik output normal maupun error) ke dalam file log backup.log.

Gambar 13. Konfigurasi crontab (cron job)

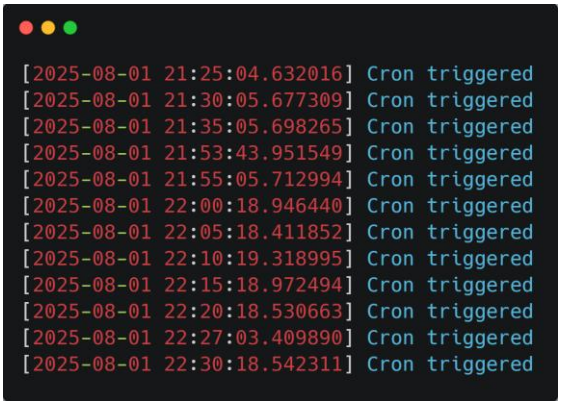
4.7. Uji Coba

Pengujian dilakukan untuk mengevaluasi keberhasilan sistem dalam menyimpan hasil pencadangan ke Google Drive secara otomatis. Gambar 14 menunjukkan salah satu log proses berhasil mengekspor database dan mengunggah hasilnya ke Google Drive. Secara umum, proses backup dan sinkronisasi berjalan dengan sukses.



Gambar 14. Log backup database dan Log upload ke Google Drive

Selama 40 menit uji coba sistem backup otomatis, dilakukan proses pencadangan secara berkala menggunakan skrip yang dijalankan melalui cron job pada Gambar 15. Tujuan uji coba ini adalah untuk mengevaluasi konsistensi, durasi, serta keberhasilan proses backup dan upload file ke Google Drive.



Gambar 15. Proses pencadangan secara berkala melalui cron job

Hasil dari uji coba tersebut dirangkum dalam tabel di bawah ini, yang merupakan ringkasan log backup secara otomatis. Tabel ini mencatat informasi penting dari setiap proses backup yang berlangsung selama pengujian, meliputi

a. Waktu Mulai Backup : waktu ketika proses backup dimulai.
b. Nama File Output : nama file hasil backup yang dihasilkan.
c. Ukuran : ukuran file backup yang dihasilkan.
d. Durasi : waktu yang dibutuhkan untuk menyelesaikan proses backup hingga upload ke Google Drive.
e. Status Upload : status keberhasilan pengunggahan ke Google Drive.
f. ID File Google Drive : ID unik file yang tersimpan di Google Drive, yang dapat digunakan untuk mengakses file tersebut.

Tabel 1. Ringkasan log backup secara otomatis

Waktu Mulai Backup	Nama File Output	Ukuran	Durasi	Status Upload	ID File Google Drive
2025-08-01 21:55:05	test_backup_2025-08-01_2155.sql	10.29 KB	6.26s	✓	13iaUyaWixTd4wThgGxGwnSmomMeM0Lgz
2025-08-01 22:00:18	test_backup_2025-08-01_2200.sql	23.15 KB	10.04s	✓	1X9IGaTz_UK1nhki7f8N9zhA5dNLFO0TO
2025-08-01 22:05:18	test_backup_2025-08-01_2205.sql	33.69 KB	13.06s	✓	1MklC_7UwLj-mVThvDA5MgQjJGYxrhwl9
2025-08-01 22:10:19	test_backup_2025-08-01_2210.sql	33.69 KB	15.17s	✓	1UYMrCq5IEI0lndq7CDQj_Ty1CNf7ihNE
2025-08-01 22:15:18	test_backup_2025-08-01_2215.sql	33.69 KB	18.54s	✓	17WZf2IkTD85rciiMpsiyqYdzyeY3xI4I
2025-08-01 22:20:18	test_backup_2025-08-01_2220.sql	33.69 KB	20.62s	✓	1yj_dPK3vDPAsGhblVL1T9RFI6w6Ulk3F
2025-08-01 22:27:03	test_backup_2025-08-01_2227.sql	33.69 KB	21.70s	✓	1g2lMGMIwji1bd4MqHKSWLzYAq1w8nuCd
2025-08-01 22:30:18	test_backup_2025-08-01_2230.sql	45.41 KB	29.23s	✓	1Yi_1Woq6SjVkXmyKUAJ8A0p4vReKo0uQ

Berdasarkan log backup yang dilakukan sebanyak delapan kali dalam rentang waktu sekitar 40 menit, seluruh proses berhasil dengan status 100% tanpa kegagalan. Waktu eksekusi backup cenderung meningkat seiring bertambahnya jumlah file yang diunggah dan diperbarui, dengan durasi tercepat 6,26 detik pada backup pertama berukuran 10,29 KB dan durasi terlama 29,23 detik pada backup terakhir berukuran 45,41 KB. Pertumbuhan ukuran file backup terbaru menunjukkan kenaikan signifikan, dari 10,29 KB pada backup awal menjadi 45,41 KB pada backup kedelapan, dengan pola

lonjakan awal, kemudian stabil di 33,69 KB pada beberapa backup, lalu meningkat tajam pada akhir proses. Setiap kali backup, sistem tidak hanya mengunggah file baru, tetapi juga memperbarui semua file lama, sehingga menambah durasi proses dan konsumsi bandwidth. Meskipun integrasi Google Drive berjalan lancar dan pencatatan log sangat jelas, terdapat potensi masalah berupa meningkatnya durasi backup, risiko keamanan akibat penggunaan password MySQL di command line, serta lonjakan ukuran file yang perlu dianalisis lebih lanjut. Untuk peningkatan, disarankan penggunaan metode incremental backup agar tidak perlu memperbarui semua file lama, pengamanan kredensial dengan metode yang lebih aman, pemantauan lonjakan ukuran file, pengarsipan file lama tanpa update berulang, dan evaluasi interval backup untuk efisiensi sumber daya.

5. KESIMPULAN DAN SARAN

Hasil penelitian menunjukkan bahwa sistem cloud backup aplikasi kepuasan pasien berbasis Docker dengan integrasi Google Drive API berhasil diimplementasikan dan berfungsi secara optimal. Proses pencadangan database MySQL yang berjalan dalam container Docker dapat dilakukan secara otomatis menggunakan skrip Python yang dijadwalkan melalui Cron Job, dengan tingkat keberhasilan 100% selama uji coba delapan kali pencadangan dalam rentang waktu 40 menit. Durasi proses backup bervariasi antara 6,26 detik hingga 29,23 detik, dipengaruhi oleh pertumbuhan ukuran file dan pembaruan seluruh file lama di Google Drive. Pencatatan log yang detail memastikan proses dapat dipantau dengan baik.

Meskipun sistem telah memenuhi tujuan penelitian dengan menyediakan mekanisme pencadangan otomatis yang portabel, terintegrasi, dan bebas intervensi manual, terdapat peluang pengembangan lebih lanjut. Rekomendasi yang dapat diterapkan meliputi penerapan metode incremental backup untuk mengurangi durasi dan konsumsi bandwidth, peningkatan keamanan kredensial dengan metode penyimpanan yang terenkripsi atau manajemen rahasia, pengarsipan file lama untuk menghindari pembaruan berulang, pemantauan pertumbuhan ukuran file untuk mendeteksi anomali, serta penyesuaian interval backup agar penggunaan sumber daya lebih efisien.

DAFTAR PUSTAKA

[1] A. F. Farhans Izzudin, “RANCANG BANGUN APLIKASI LAYANAN KEPUASAN PASIEN DI PUSKESMAS SUKODADI LAMONGAN BERBASIS WEBSITE,” no. Ci, 2015.

[2] Y. Al-Issa, M. A. Ottom, and A. Tamrawi, “EHealth *Cloud* Security Challenges: A Survey,” *J Healthc Eng*, vol. 2019, 2019, doi: 10.1155/2019/7516035.

[3] J. T. Santoso, “Komputasi awan,” *Computing*, pp. 0–297, 2023.

[4] Y. S. Butar-butar, “Pengelolaan Data dalam Lingkungan *Cloud Computing* : Ulasan Jurnal Pengelolaan Data dalam Lingkungan *Cloud Computing* Pengantar,” no. December, pp. 0–5, 2023.

[5] D. Merkel, “Docker : Lightweight Linux *Container*s for Consistent Development and Deployment Docker : a Little Background Under the Hood,” *Linux Journal*, vol. 2014, no. 239, pp. 2–7, 2014.

[6] I Putu Eka Indrawan, Gde Iwan Setiawan, and Adelia A’fa Nafasha, “Analisis Perbandingan Biaya Dan Serverless *Computing* Pada *Google Cloud Platform*,” *Jurnal Manajemen dan Teknologi Informasi*, vol. 14, no. 1, pp. 1–9, 2024, doi: 10.59819/jmti.v14i1.3668.

[7] T. H. Subaktiono, “Trend Keamanan *Cloud Computing*,” *Universitar Komputer Indonesia*, no. April, pp. 1–12, 2023.

[8] Setiawan Wawan, ”ANALISA LAYANAN *CLOUD COMPUTING* DI ERA DIGITAL,” Universitas Muhammadiyah Tangerang, No1. Maret, 2022.

[9] A. R. Ekaputra and A. S. Affandi, “Pemanfaatan layanan *cloud computing* dan docker *container* untuk meningkatkan kinerja aplikasi web,” *Journal of Information System and Application Development*, vol. 1, no. 2, pp. 138–147, Sep. 2023, doi: 10.26905/jisad.v1i2.11084.

[10] A. A’fa Nafasha, I. Putu, E. Indrawan, and G. I. Setiawan, “ANALISIS PERBANDINGAN BIAYA DAN SERVERLESS *COMPUTING* PADA GOOGLE *CLOUD* PLATFORM,” *Jurnal Manajemen dan Teknologi Informasi (JMTI)*, vol. 14, pp. 1–09, doi: 10.59819.

[11] I. Uliyah Sari, M. Tahir, and E. Vincentius, “Analisis Komparatif Layanan *Cloud* : Microsoft Azure, Aws, Dan Google *Cloud* Platform (GCP),” *Sains dan Teknologi*, vol. 6, no. 2.

[12] I. Sholihah and C. Darujati, “Sistem Replikasi Basis Data berdasarkan MySQL menggunakan *Container* Docker,” *Majalah Ilmiah Teknologi Elektro*, vol. 21, no. 2, p. 209, Dec. 2022, doi: 10.24843/mite.2022.v21i02.p08.

[13] A. Yulyan Chandra *et al.*, “Perancangan Sistem Automasi Backup Konfigurasi Perangkat Pada Internet Service Provider,” 2024.

[14] E. Ramadhani, “Desain E-Health: Sistem Keamanan Aplikasi E-health Berbasis *Cloud Computing* Menggunakan Metode Single Sign On,” 2015