

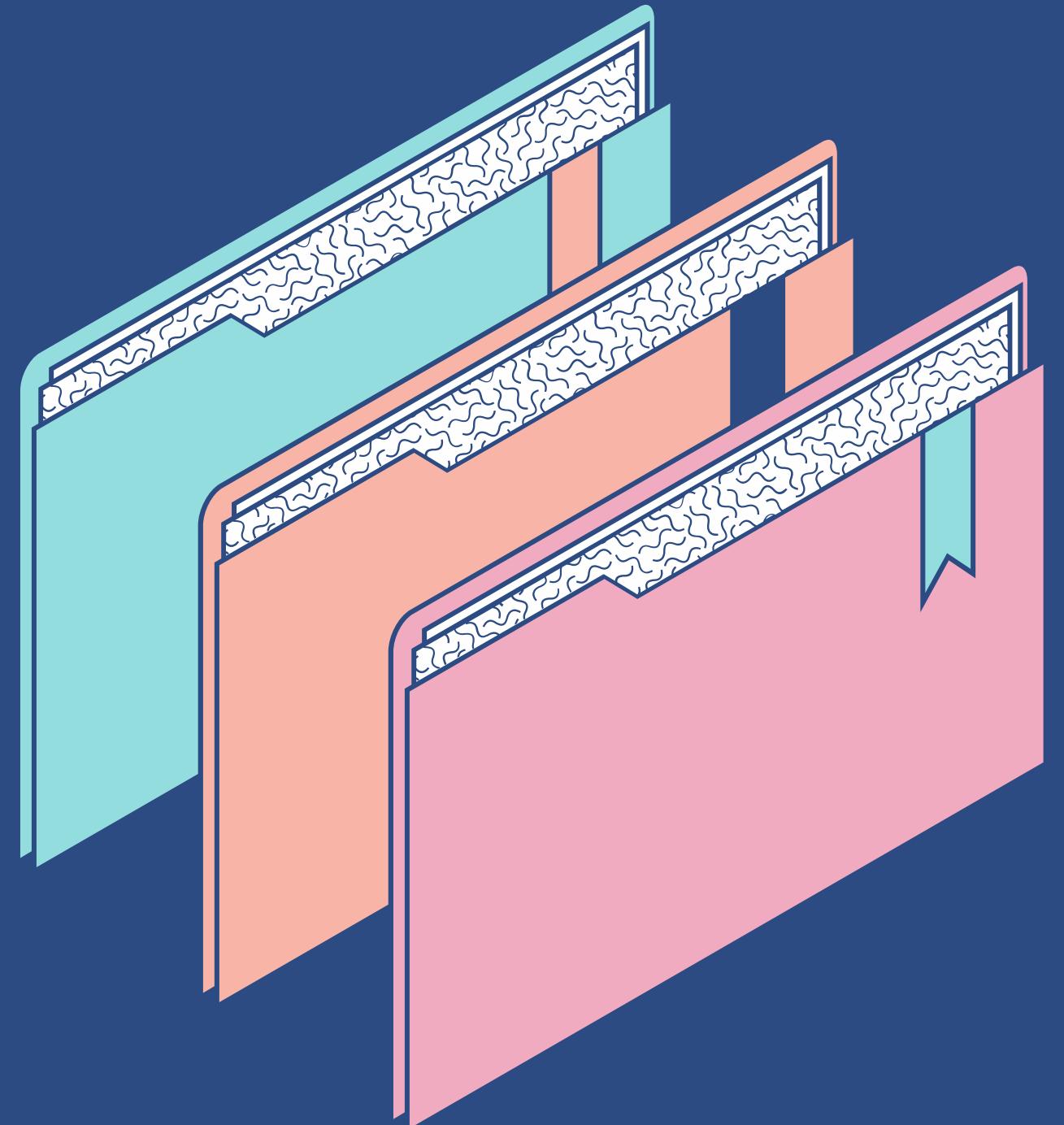


PENERAPAN CI/CD
DISASTER MANAGEMENT CYCLE DALAM
KONTEKS DEVOPS

RupiahNOW Konverter

Website konversi nilai tukar mata uang
secara real time

Link Video Presentasi : https://www.youtube.com/watch?v=V8KED9El_yo



KELOMPOK 3



AULIA VIKA RAHMAN
(2208107010001)



SADINAL MUFTI
(2208107010007)



Pendahuluan

Latar Belakang:

- Kebutuhan akan informasi nilai tukar mata uang yang cepat dan akurat.
- Aplikasi RupiahNow sebagai solusi konversi mata uang real-time.

Rumusan Masalah:

- Menjaga ketersediaan dan performa aplikasi.
- Strategi CI/CD, Backup & Recovery, dan Disaster Management.



Tujuan

Tujuan Utama:

- Memastikan aplikasi berjalan lancar meskipun terjadi gangguan.

Tujuan Khusus:

- Implementasi CI/CD untuk pengiriman kode cepat dan aman.
- Strategi backup data untuk menjaga integritas.
- Disaster Management Cycle untuk kesiapan menghadapi bencana.

Metodologi

1

DESAIN SISTEM

Sistem aplikasi
RupiahNow

2

IMPLEMENTASI CI/CD

- Build
- Test
- Push
- Deploy

3

STRATEGI BACKUP & RECOVERY

menyimpan salinan
data dan
pemulihan

4

DISASTER MANAGEMENT CYCLE

- Mitigasi
- Kesiapsiagaan
- Respons
- Pemulihan

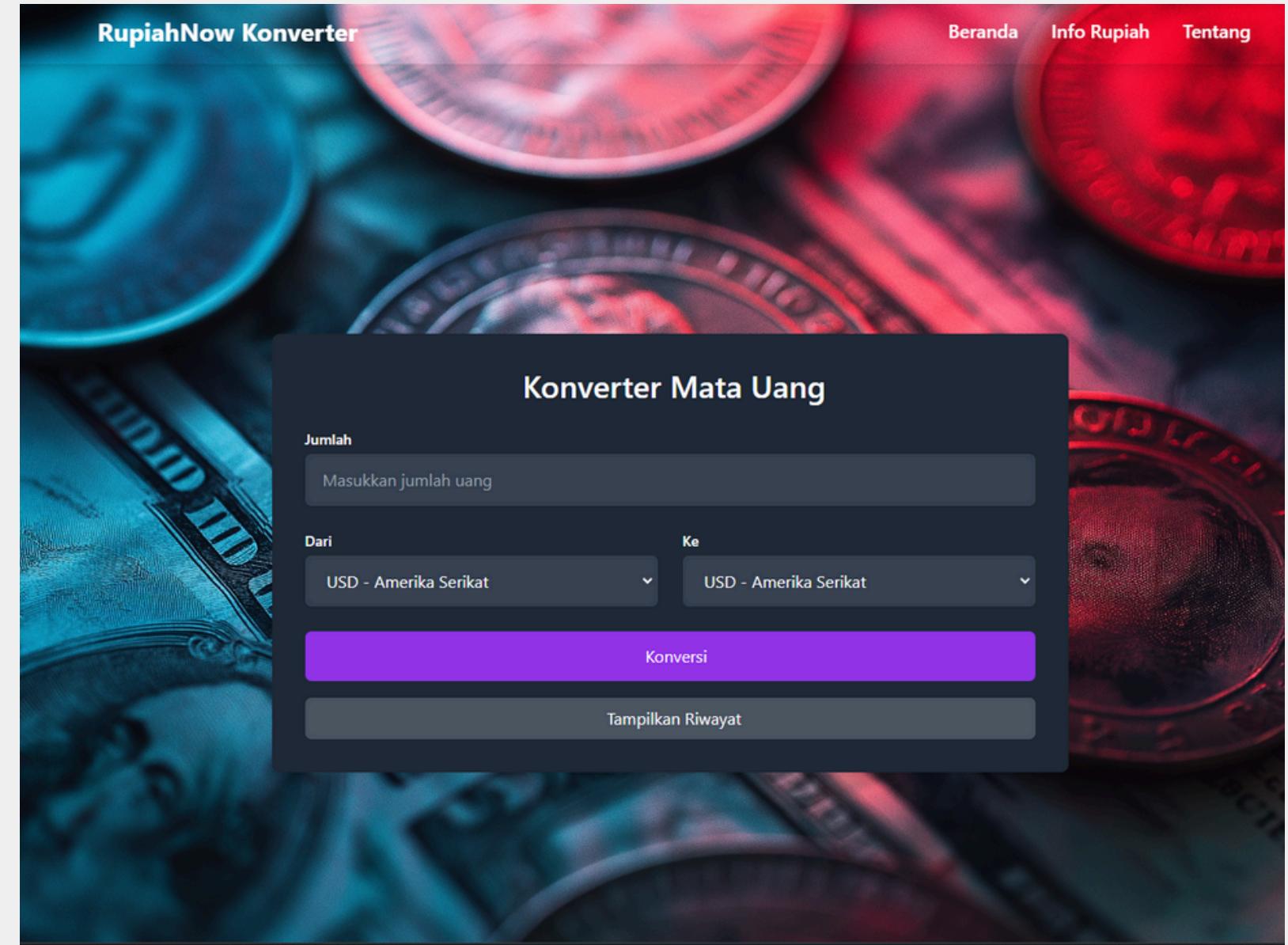
1. Desain Sistem

CI/CD (Continuous Integration/Continuous Deployment):

Setiap perubahan kode yang dikirimkan ke GitHub melalui pipeline CI/CD secara otomatis diuji dan dideploy ke lingkungan produksi setelah lolos pengujian. Ini memastikan pengembangan yang cepat, pengujian otomatis, dan pengiriman fitur tanpa gangguan.

Tools :

- Github
- Docker
- Google Cloud Platform (GCP)



2. Implementasi CI/CD

- **Build:** Aplikasi dibangun dan dikemas dalam Docker image.
- **Test:** Kode diuji secara otomatis untuk memastikan bahwa tidak ada kesalahan yang muncul.
- **Push:** Docker image yang telah teruji didorong ke DockerHub.
- **Deploy:** Aplikasi kemudian dideploy secara otomatis ke server di GCP.

The screenshot shows a CI/CD pipeline interface with three main sections: Build, Test, and Deployment.

- Build:** Step 1: Run go build -v ./...
Step 4: internal/goarch
Step 5: internal/coverage/rtcov
Step 6: internal/unsafeheader
Step 7: internal/cpu
Step 8: internal/goexperiment
Step 9: internal/goos
Step 10: internal/abi
Step 11: runtime/internal/atomic
- Test:** Step 1: Run go test -v ./...
Step 2: go test -v ./...
Step 3: shell: /usr/bin/bash -e {0}
Step 4: ? github.com/auliavika/RupiahNOW [no test files]
- Deployment:**
 - > Login to Docker Hub
 - > Build and push
 - > Post Build and push
 - > Post Login to Docker Hub

3.Strategi Backup & Recovery

Jenis Backup:

- Full Backup
- Incremental Backup
- Differential Backup

Proses Backup:

- Penjadwalan Backup Otomatis
- Backup Data Aplikasi dan Konfigurasi

Proses Recovery:

- Pemulihan data dari cloud.
- Pemulihan Server atau Infrastruktur
- Rollback kode aplikasi jika terjadi kesalahan.

```
● ● ●  
# Melihat riwayat commit  
git log  
  
# Menentukan commit ID dari versi yang ingin dipulihkan  
git checkout <commit-id>  
  
# Mengembalikan perubahan ke versi yang diinginkan  
git push origin HEAD:main  
  
# Deploy versi rollback menggunakan Docker  
docker build -t username/rupiahnow:rollback .  
docker push username/rupiahnow:rollback
```

DISASTER MANAGEMENT CYCLE

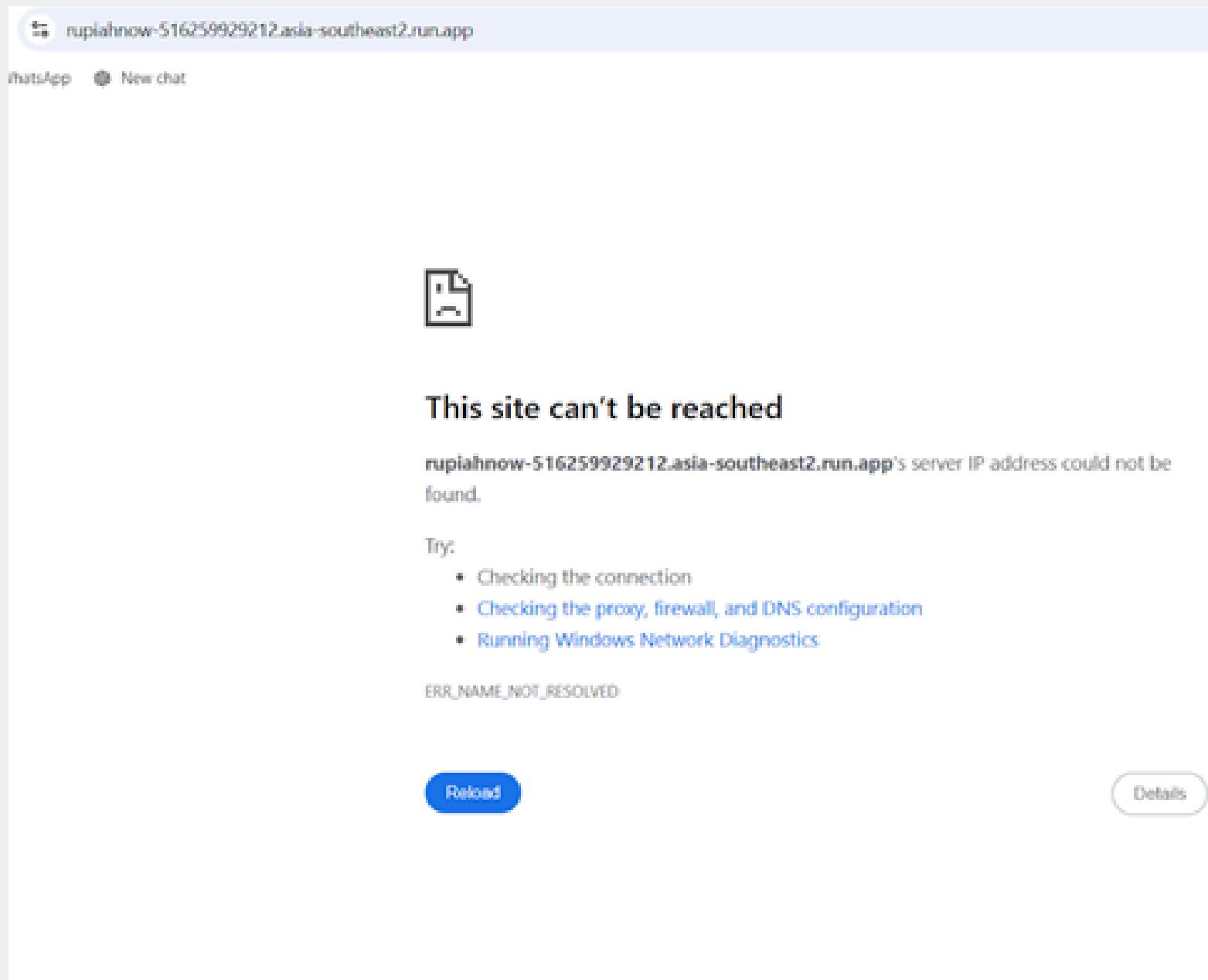
1. **Mitigasi:** Monitoring dan alerting dilakukan untuk mendeteksi potensi masalah pada server.
2. **Kesiapsiagaan:** Pengujian failover dan pemulihan dilakukan secara berkala untuk memastikan sistem siap menghadapi bencana.
3. **Respons:** Pada saat terjadi masalah, failover otomatis mengalihkan traffic ke server lain, dan pengembangan aplikasi diteruskan tanpa gangguan.
4. **Pemulihan:** Data yang hilang atau rusak dapat dipulihkan dengan menggunakan snapshot backup yang disimpan di Google Cloud.





Hasil dan Simulasi Bencana

1. Simulasi Downtime



Pada simulasi pertama, masalah yang muncul disebabkan oleh gangguan jaringan atau koneksi internet yang menghalangi aplikasi untuk mengakses layanan API eksternal yang menyediakan data nilai tukar

Dampak yang ditimbulkan :

- Tidak dapat mengakses aplikasi
- Kegagalan menampilkan informasi
- Penurunan pengalaman pengguna

2. Simulasi Kerusakan Server

! Failed: b2c2e0f2-5a76-407c-96cb-7310e5b7	
Started on Dec 7, 2024, 11:16:23 PM	
Steps	Duration
! Build Summary 3 Steps	00:05:19
✓ 0: Build build -no-cache -t asia-southeast2-docker.pkg.dev/note...	00:00:36
✓ 1: Push push asia-southeast2-docker.pkg.dev/noted-point-4410...	00:00:25
! 2: Deploy gcloud run services update rupiahnow --platform=mana...	00:04:13

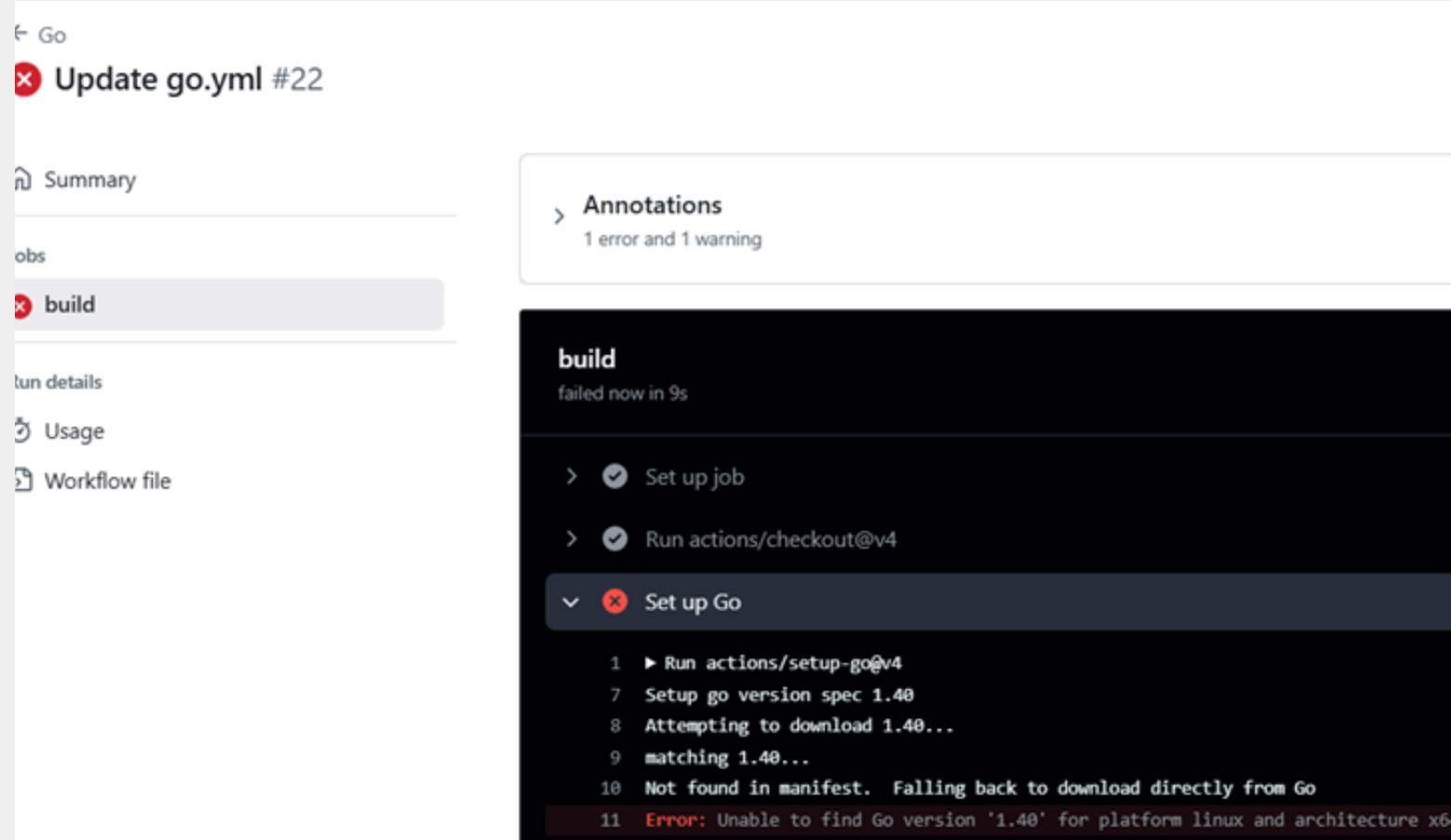
Step #2 - "Deploy": ERROR: (gcloud.run.services.update) Revision 'rupiahnow-00009-mrw' is not ready and cannot serve traffic. user-provided container failed to start and listen on the port defined provided by the PORT=8080 environment variable within the allocated timeout. This can happen when the container port is misconfigured or if the timeout is too short. The health check timeout can be extended. Logs for this revision might contain more information.

Pada simulasi ini, masalah yang muncul disebabkan oleh ketidaksesuaian antara port yang digunakan oleh aplikasi di dalam kode dan port yang diatur pada konfigurasi server di Google Cloud Platform (GCP).

Dampak yang ditimbulkan :

- Kegagalan akses aplikasi
- Layanan konversi tidak dapat diakses

3. Simulasi Kegagalan Aplikasi



Pada simulasi ini, aplikasi mengalami kegagalan selama proses build karena ketidaksesuaian versi Go yang digunakan dalam pengembangan aplikasi dan versi yang diatur pada environment saat build dijalankan.

Dampak yang ditimbulkan:

- Build Gagal
- Gangguan dalam Pengiriman Pembaruan
- Ketidaktersediaan Aplikasi
- Penurunan Kepercayaan Pengguna



Strategi Mitigasi

a. Mitigasi Downtime

- Penggunaan API Cadangan
- Caching Data Nilai Tukar
- Pemilihan Infrastruktur Cloud yang Tanggu

b. Mitigasi Kerusakan Server

- Penggunaan Konfigurasi yang Konsisten
- Pemeliharaan dan Monitoring Infrastruktur

c. Mitigasi Kegagalan Aplikasi

- Penggunaan Versi Golang yang Konsisten
- CI/CD untuk Build dan Pengujian Otomatis



Kesiapsiagaan

a. Kesiapsiagaan untuk Downtime

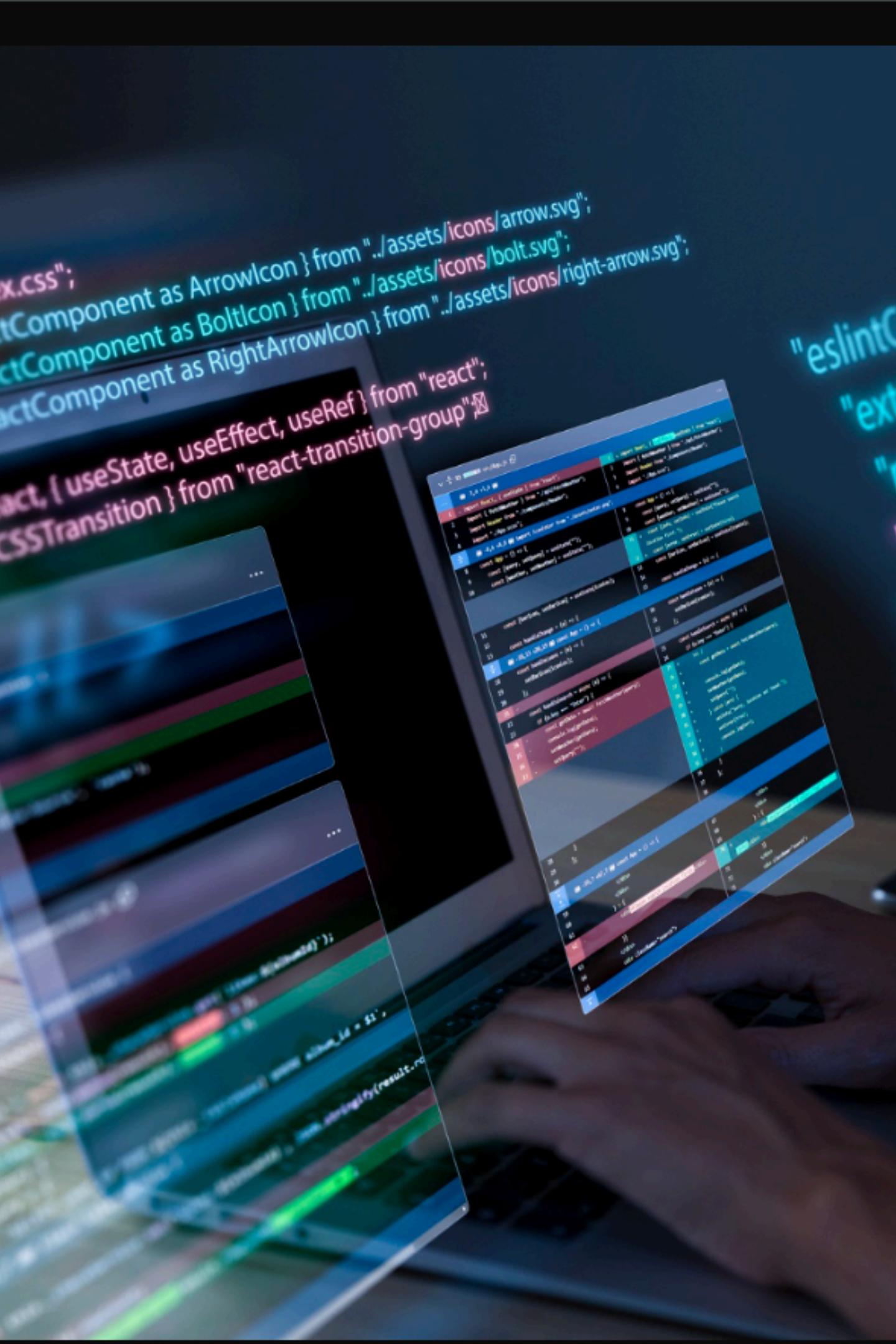
- Pemantauan Sistem Secara Real-time
- Prosedur Pemulihan Darurat

b. Kesiapsiagaan untuk Kerusakan Server

- Pemantauan Konfigurasi Secara Real-time

c. Kesiapsiagaan untuk Kegagalan Aplikasi

- Pengujian Integrasi dan Ketergantungan
- Dokumentasi dan Pelatihan Tim



Respons

a. Respons Downtime

- Pengalihan ke API Cadangan dan Caching
- Notifikasi Kepada Pengguna

b. Respons Kerusakan Server

- Perbaikan Konfigurasi Port
- Penggantian Konfigurasi di GCP

c. Respons Kegagalan Aplikasi

- Penyelesaian Isu Build
- Hotfix dan Penerapan Patch



Pemulihan

a. Pemulihan untuk Downtime

- Pemulihan API dan Caching
- Verifikasi Data dan Pengujian Aplikasi

b. Pemulihan untuk Kerusakan Server

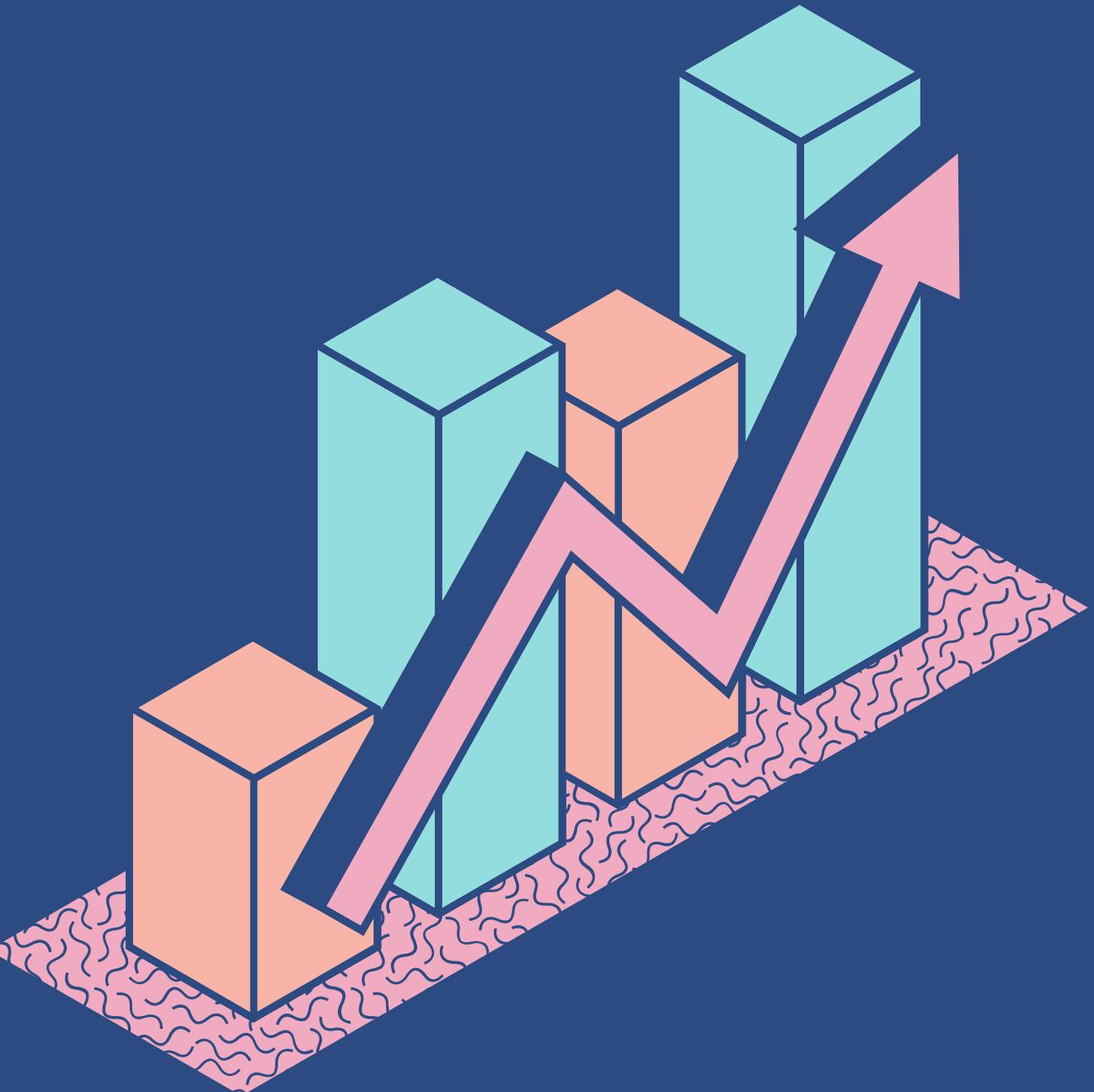
- Perbaikan dan Pengujian Aplikasi
- Verifikasi Pengaturan Server di GCP

c. Pemulihan untuk Kegagalan Aplikasi

- Penyelesaian Bug dan Penerapan Fixes
- Pengujian Ulang Aplikasi dan Penerapan Versi yang Tepat

KESIMPULAN

Analisis ini menegaskan pentingnya DevOps, CI/CD, Backup & Recovery, dan Disaster Management Cycle dalam menjaga operasional aplikasi RupiahNow. Pendekatan ini memastikan layanan tetap andal saat menghadapi downtime, kerusakan server, atau kegagalan aplikasi. Strategi mitigasi, kesiapsiagaan, respons cepat, dan pemulihan efektif meningkatkan keandalan aplikasi serta kepercayaan pengguna.



THANK YOU