Deskripsi Proyek dan Tugas

Modul ini adalah **Deployment Automation dan CI/CD**.

Anda berperan sebagai seorang **engineer DevOps** di techno Cooporate., sebuah perusahaan teknologi yang berkembang pesat dan baru saja meningkatkan jumlah tim pengembangnya dari 10 menjadi 50 orang. Seiring dengan pertumbuhan ini, jumlah proyek juga meningkat secara signifikan, sehingga menimbulkan tantangan dalam manajemen dan koordinasi proyek. Tim mengalami kesulitan dalam melacak tugas, tenggat waktu, dan progres di berbagai proyek, yang mengakibatkan banyak tenggat waktu terlewat dan alur kerja menjadi tidak efisien.

Untuk mengatasi permasalahan tersebut, CTO memutuskan untuk menerapkan sebuah alat manajemen proyek. Namun, karena adanya persyaratan kepatuhan yang ketat, perusahaan tidak dapat menggunakan solusi SaaS publik seperti Jira atau Trello. Membangun aplikasi manajemen proyek khusus dari awal juga tidak memungkinkan karena membutuhkan waktu yang lama dan biaya yang besar.

Setelah mengevaluasi berbagai pilihan, CTO dan tim pengembang memutuskan untuk menggunakan alat manajemen proyek open-source yang di-host sendiri (self-hosted). Tugas Anda adalah memastikan aplikasi tersebut dapat dideploy dan dipantau secara efektif agar tim dapat bekerja tanpa gangguan. Selain itu, Anda juga harus memastikan bahwa solusi yang digunakan aman dan sesuai dengan ketentuan kepatuhan, karena perusahaan menangani data pengguna yang bersifat sensitif.

Tugas Peserta

Pada modul ini, peserta diminta untuk menyiapkan dan menjalankan sistem **Deployment Automation dan CI/CD** menggunakan layanan AWS dan GitHub. Berikut langkah-langkah yang harus dikerjakan:

1. Baca seluruh modul dan dokumentasi dengan teliti agar memahami alur pengerjaan.
2. Siapkan akun **GitHub** dan pastikan bisa digunakan untuk push kode dan mengatur GitHub Actions.
3. Login ke **AWS Academy Lab** untuk membuat dan mengonfigurasi seluruh layanan AWS.

4. Pahami terlebih dahulu bagian **Detail Aplikasi** agar tahu aplikasi apa saja yang akan dideploy dan bagaimana cara kerjanya.
5. Buat **VPC** sesuai instruksi pada bagian jaringan untuk memisahkan jaringan publik dan privat.
6. Atur **security group** agar akses ke aplikasi, database, dan file system aman.
7. Buat **database relasional** sesuai ketentuan yang diberikan.
8. Konfigurasikan **file system (EFS)** untuk menyimpan file yang diunggah oleh aplikasi.
9. Simpan data sensitif seperti username dan password database di **SSM Parameter Store**, bukan di kode.
10. Buat **container registry (ECR)** untuk menyimpan image aplikasi.
11. Konfigurasikan **Application Load Balancer** dan **Auto Scaling** agar aplikasi bisa diakses dan tetap berjalan meskipun trafik meningkat.
12. Buat **ECS Cluster** dan jalankan aplikasi menggunakan **ECS Fargate**.
13. Konfigurasikan **GitHub Actions** sebagai pipeline **CI/CD** untuk proses build dan deploy otomatis.
14. Lakukan **pengecekan akhir** untuk memastikan aplikasi berjalan dengan baik dan dapat diakses.

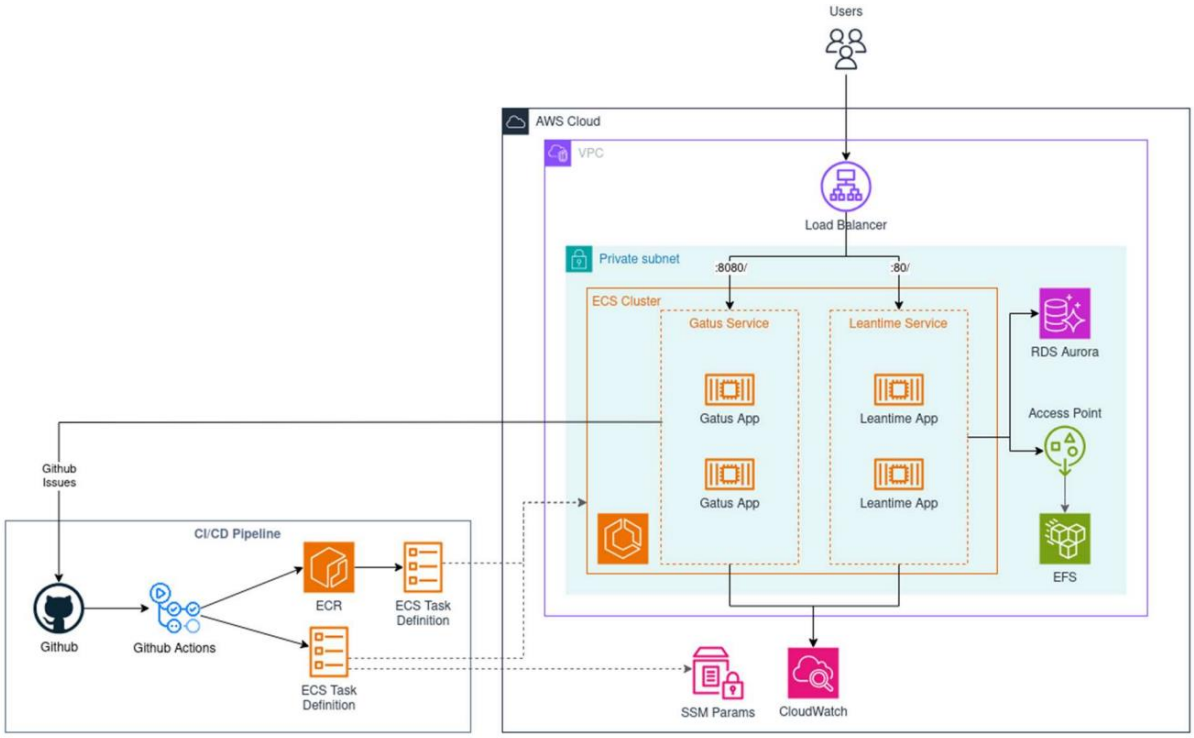
Referensi

Untuk membantu peserta memahami dan mengerjakan modul ini, berikut beberapa dokumentasi resmi yang dapat dijadikan acuan:

1. Dokumentasi **aplikasi Leantime** dapat diakses melalui situs resmi Leantime.
   https://docs.leantime.io/
2. Dokumentasi **aplikasi Gatus** dapat dipelajari melalui website resmi Gatus maupun halaman GitHub resminya.
   https://gatus.io/docs | https://github.com/TwiN/gatus?tab=readme-ov-file#table-of-contents
3. Dokumentasi mengenai **Amazon VPC** dapat digunakan sebagai panduan dalam membangun dan mengatur jaringan di AWS.
   https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html
4. Dokumentasi **Amazon RDS (Aurora)** dapat digunakan untuk memahami cara membuat dan mengelola database.
   https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/CHAP_A

5. Dokumentasi **Amazon EFS** dapat digunakan sebagai referensi dalam pembuatan dan pengaturan file system.
   https://docs.aws.amazon.com/efs/latest/ug/whatisefs.html
6. Dokumentasi **Amazon ECS** dapat digunakan untuk memahami cara menjalankan aplikasi berbasis container.
   https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.ht
7. Dokumentasi **Amazon ECR** dapat digunakan sebagai panduan penyimpanan image container.
   https://docs.aws.amazon.com/AmazonECR/latest/userguide/what-is-ecr.html
8. Dokumentasi **SSM Parameter Store** dapat digunakan sebagai referensi dalam menyimpan dan mengelola konfigurasi serta data sensitif.
   https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-parameter-store.html
9. Dokumentasi **Application Load Balancer (ALB)** dapat digunakan untuk memahami cara mengatur distribusi trafik aplikasi.
   https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introducti
10. Dokumentasi **GitHub Actions** dapat digunakan sebagai panduan dalam membuat workflow CI/CD.
    https://docs.github.com/en/actions/writing-workflows/quickstart
11. Dokumentasi **GitHub** https://github.com/lksccsda/devops-lksccsda-26

Arsitektur



Detail Aplikasi

Pada modul ini terdapat **dua aplikasi** yang harus dideploy. Keduanya merupakan **aplikasi web open-source**. Aplikasi pertama adalah **Leantime**, yaitu aplikasi yang digunakan untuk **manajemen proyek**. Aplikasi kedua adalah **Gatus**, yaitu **dashboard monitoring sistem** yang berfungsi untuk memantau kondisi halaman web dan mengirimkan notifikasi jika terjadi gangguan. Tugas awal yang harus dilakukan adalah **mengonfigurasi aplikasi**, baik melalui **environment variable** maupun melalui **file konfigurasi khusus** sesuai kebutuhan masing-masing aplikasi. Setelah konfigurasi aplikasi selesai, tugas berikutnya adalah **membuat dan mengatur ECS task definition**. Task definition ini berfungsi untuk memastikan setiap aplikasi dapat dijalankan dengan benar sebagai **ECS Service** di AWS.

Aplikasi Leantime

Aplikasi Leantime memerlukan konfigurasi **environment variable** yang tepat agar dapat berfungsi dengan baik di lingkungan deployment Anda. Contoh file environment tersedia di repository yang berada di direktori `leantime/`. Konfigurasikan aplikasi dengan mengatur **URL aplikasi** dan **port aplikasi** ke **8080** agar sesuai dengan konfigurasi container Anda. Atur **sitename** menjadi **LKS Leantime** untuk memberi identitas aplikasi dengan benar. Konfigurasikan **timezone default** ke **Asia/Jakarta** untuk memastikan semua fitur berbasis waktu ditampilkan dengan benar kepada pengguna.

Anda diperbolehkan untuk menyesuaikan warna atau tema sesuai preferensi, namun pastikan semua penyesuaian tersebut dapat berjalan dengan baik tanpa merusak fungsi aplikasi. Selama tahap pengembangan, Anda dapat mengaktifkan **mode debug** untuk membantu pemecahan masalah, tetapi Anda **harus menonaktifkannya** sebelum menyelesaikan proses deployment agar aplikasi siap digunakan di lingkungan produksi dan tetap aman.

Definisikan **ECS task definition** untuk aplikasi Leantime menggunakan file yang tersedia di repository pada `leantime/task-definition.json`. Lengkapi konfigurasi yang masih kosong sesuai dengan penanda pada file tersebut. Atur **family name** menjadi `lks-leantime-app`, dan pada bagian **container definitions**, map **container port** dan **host port** ke **8080** atau sesuaikan dengan port aplikasi yang ditentukan pada environment variable.

Konfigurasikan **log driver** menjadi `awslogs` dan atur **log group** ke `/ecs/lks/leantime-app` agar log dikirim ke **AWS CloudWatch**. Gunakan **AWS VPC** sebagai **network mode**, batasi **required compatibilities** pada task definition hanya untuk **Fargate**, dan alokasikan **1024 unit CPU** serta **2048 MB memori**.

Tambahkan seluruh **environment variable** yang diperlukan pada properti **environment** di dalam container definition. Setiap environment variable yang berisi nilai sensitif seperti **password, secret, dan kredensial database** harus disimpan dan diambil dari **SSM Parameter Store** untuk mencegah kerentanan keamanan yang dapat menyebabkan pengurangan nilai (baca bagian **Service Details – SSM Parameter Store** untuk pengaturan parameter store).

Definisikan **volume** berdasarkan **EFS access point** dan hubungkan ke **container mount point** dengan pengaturan **read-only dinonaktifkan** serta mengikuti konfigurasi yang telah ditentukan.

EFS Access Point	Container Path	Description
/public-files	/var/www/html/public/userfiles	Store public files such as the logo, etc
/userfiles	/var/www/html/userfiles	Store the user uploaded files
/plugins	/var/www/html/app/Plugins	Store the plugin for the application

Gatus

Aplikasi ini tidak memerlukan **environment variable**, melainkan menggunakan satu file **YAML** untuk konfigurasi. File tersebut berada di `gatus/config/config.yaml`. Langkah awal adalah melakukan konfigurasi **alerting provider** menggunakan **GitHub Issues** di dalam repository. **JANGAN** menyimpan URL dan token GitHub yang digunakan untuk mengirim alert secara langsung. Hal ini harus dilakukan melalui **environment variable** pada proses **CI/CD** menggunakan **GitHub Actions**. Kegagalan dalam melakukan hal ini dapat menyebabkan kerentanan keamanan pada aplikasi dan mengakibatkan hilangnya seluruh poin yang diperoleh.

Buat dua **endpoint**, yaitu untuk **URL aplikasi Leantime** dan **URL Gatus**. Atur interval pengecekan URL Leantime menjadi **30 detik**, dan konfigurasikan kondisi untuk menentukan kesehatan endpoint dengan **status code 200**, serta waktu respons harus **di bawah 300 ms**. Untuk URL Gatus, atur interval menjadi **1 menit**, dan konfigurasikan kondisi kesehatan endpoint dengan **status code 200** serta waktu respons **di bawah 500 ms**.

Atur **alerting provider** ke **GitHub** dengan **failure threshold 3** dan **success threshold 5**. Aktifkan pengaturan **send on resolved** agar issue dapat ditutup secara otomatis ketika masalah telah terselesaikan. Anda dapat menggunakan deskripsi default yang tersedia di repository atau membuat deskripsi khusus sendiri.

Untuk mendefinisikan **ECS task definition** aplikasi Gatus, gunakan task definition dasar yang tersedia di repository pada `gatus/task-definition.json`. Lengkapi konfigurasi yang masih kosong sesuai penanda di dalam file tersebut. Atur **family name** menjadi `lks-gatus-app` dan konfigurasikan **container definitions** dengan memetakan **container port** dan **host port** ke **8080**.

Selain itu, atur **log driver** ke `awslogs` dan konfigurasikan **log group** menjadi `/ecs/lks/gatus-app` agar log tersimpan dengan baik di **AWS CloudWatch**. Atur **network mode** menggunakan **AWS VPC**, tentukan bahwa properti **required compatibilities** hanya untuk **Fargate**, serta alokasikan **512 unit CPU** dan **1024 MB memori** untuk task tersebut.

Detail Layanan

Jaringan (Network)

Siapkan infrastruktur jaringan sebelum melakukan deployment aplikasi pada region yang telah ditentukan dengan CIDR **10.100.0.0/16**. Konfigurasikan **VPC** dengan spesifikasi berikut:

- Nama VPC harus **lks-pm-vpc**;
- Subnet publik dimulai dengan CIDR **10.100.0.0/24** dengan minimal **2 availability zone**;
- Subnet privat dimulai dengan CIDR **10.100.10.0/24** dengan minimal **2 availability zone**;
- Nama **Internet Gateway** adalah **lks-pm-igw**;
- Gunakan **route table terpisah** untuk rute publik dan privat. Beri nama route table tersebut masing-masing **lks-pm-public-rt** dan **lks-pm-private-rt**;
- Pasang **1 NAT Gateway** dan beri nama **lks-pm-natgw**.

Keamanan (Security)

Keamanan merupakan aspek paling penting yang harus diperhatikan dalam membangun infrastruktur. Buat **security group** untuk instance aplikasi yang hanya mengizinkan trafik masuk (**inbound**) dari **Application Load Balancer (ALB)**. Hal ini membatasi akses ke layanan sehingga hanya dapat diakses melalui ALB, yang berfungsi sebagai gerbang untuk trafik masuk.

Application Load Balancer (ALB) memiliki **security group terpisah** yang hanya membuka port **HTTP 80 dan 8080** dari jaringan publik.

Amankan **database** dan **file system** dengan mencegah akses publik, sehingga sumber daya tersebut tidak dapat diakses langsung dari internet. Koneksi ke database dan file system hanya diizinkan dari aplikasi, sehingga akses terbatas pada sumber yang sah. Langkah-langkah ini meningkatkan tingkat keamanan lingkungan, mengurangi risiko akses tidak sah, serta menjaga integritas aplikasi dan data yang dikelola.

Database

Aplikasi menggunakan **MySQL** untuk menyimpan informasi pengguna. Buat **RDS Aurora untuk MySQL** dengan kompatibilitas versi utama **MySQL 8.0**. Gunakan **template production** dan atur **cluster identifier** menjadi **lks-project-management-cluster**. Pilih **Aurora Standard** untuk konfigurasi penyimpanan. Aktifkan **Aurora replica** dan konfigurasikan instance dengan **instance class t4g.medium** untuk mendapatkan performa dan efisiensi biaya terbaik.

Amankan database dengan mencegah akses dari jaringan publik dan menggunakan **security group** yang telah dibuat sebelumnya. Pada bagian konfigurasi monitoring, gunakan **standard database insight** dan **nonaktifkan enhanced monitoring**.

Anda juga perlu menyiapkan **username**, **password**, dan **leantime_db** sebagai **nama database awal** yang dibutuhkan oleh aplikasi. Salin dan simpan kredensial database tersebut untuk digunakan di kemudian hari.

File System

Aplikasi memerlukan **file system terpusat** untuk menyimpan seluruh file media yang diunggah oleh pengguna. Anda harus mengonfigurasikan **Elastic File System (EFS)** dengan spesifikasi berikut:

- Beri nama **lks-project-management**;
- Tipe **Regional**;
- Aktifkan **enkripsi otomatis data saat disimpan (at rest)** untuk memastikan keamanan data;
- Atur **performance mode** ke **General Purpose**;
- Atur **throughput mode** ke **enhanced, elastic**.

Amankan file system dengan mencegah akses dari jaringan publik dan menggunakan **security group** yang telah dibuat sebelumnya.

Untuk mendukung penyimpanan aset pengguna oleh aplikasi Leantime, perlu dibuat **access point** masing-masing untuk direktori **/public_userfiles**, **/userfiles**, dan **/plugins**. Gunakan **User ID** dan **Group ID** yang digunakan dalam environment variable Leantime untuk pembuatan direktori.

Atur izin akses dengan **akses penuh untuk user**, **baca dan tulis untuk grup**, serta **baca saja untuk pihak lain**, sesuai dengan prinsip **least privilege**. Access point ini digunakan oleh container dan akan di-mount ke container melalui **ECS task definition**. Bacalah bagian **Application Details – Leantime App** untuk melakukan mounting EFS ke instance aplikasi.

SSM Parameter Store

Aplikasi **Leantime** memerlukan sebuah database untuk menyimpan data di **AWS RDS**. Data konfigurasi database tersebut diambil dari **SSM Parameter Store** dengan menggunakan path **/lks/db/{{KEY}}**. Key berikut harus dibuat di dalam parameter store:

Key Name	Type	Description
Host	String	Database host/endpoint
Port	String	Database port
Name	String	Initial/default database name
Username	SecureString	Database username
Password	SecureString	Database password

Elastic Container Registry (ECR)

ECR digunakan sebagai **repository image container** untuk menyimpan aplikasi **Gatus** beserta konfigurasinya. Buat **ECR private repository** dengan nama **"lks-gatus"** dan pastikan **tag bersifat mutable**. Hal ini bertujuan agar tag **latest** pada image aplikasi selalu diperbarui.

Setiap image akan dipublikasikan secara otomatis melalui **CI/CD** ke private registry ini dengan tag berikut:

```
<account-id>.dkr.ecr.us-east-1.amazonaws.com/lks-gatus:<commit-sha>
<account-id>.dkr.ecr.us-east-1.amazonaws.com/lks-gatus:latest
```

High Availability

Untuk memastikan **ketersediaan tinggi** dan distribusi beban yang tepat pada aplikasi, Anda perlu mengonfigurasi **Target Group** untuk layanan **Gatus**. Buat target group dengan nama

lks-gatus-app-tg, atur **target type** ke **IP address**, konfigurasikan **target port** ke **8080**, dan gunakan **HTTP/1.1** sebagai **target protocol**.

Untuk pengaturan **health check**, konfigurasikan parameter berikut guna memantau kesehatan layanan:

- Gunakan **HTTP** sebagai health check protocol
- Atur **health check path** ke **root path**
- Konfigurasikan **healthy threshold** sebesar **5** dan **unhealthy threshold** sebesar **3**
- Atur **timeout period** dan **check interval** menjadi **1 menit**, serta set **200** sebagai **success code** yang menandakan target sehat

Untuk layanan **Leantime**, Anda perlu mengonfigurasi **Target Group** lain untuk mengarahkan dan memantau trafik dengan benar. Buat target group dengan nama **lks-leantime-app-tg**, atur **target type** ke **IP Address**, konfigurasikan **port** ke **8080**, dan gunakan **HTTP/1.1** sebagai **protocol**.

Untuk pengaturan health check layanan Leantime, konfigurasikan parameter berikut:

- Gunakan **HTTP** sebagai health check protocol
- Konfigurasikan **healthy threshold** sebesar **3**, **unhealthy threshold** sebesar **2**, **timeout period** selama **30 detik**, dan **check interval** selama **1 menit**

Saat mengonfigurasi **health check path** dan **success code**, Anda harus memperhatikan perilaku aplikasi Leantime, yang melakukan **redirect** dari **root path** ke **/auth/login** dengan **status code 302**.

Setelah membuat target group untuk kedua layanan, buat **Application Load Balancer** untuk mendistribusikan trafik masuk. Buat Application Load Balancer dengan nama **lks-oss-lb**, atur **scheme** ke **internet-facing** agar dapat diakses dari internet, dan konfigurasikan **IP type** ke **IPv4**. Gunakan **VPC** yang telah dibuat sebelumnya dan pilih **security group** yang sesuai untuk mengatur akses trafik.

Konfigurasikan **dua listener** untuk mengarahkan trafik ke layanan:

- Buat listener pada **HTTP port 80** yang meneruskan trafik ke target group layanan **Leantime**
- Buat listener pada **HTTP port 8080** yang meneruskan trafik ke target group layanan **Gatus**

Konfigurasi ini memungkinkan pengguna mengakses aplikasi **Leantime** melalui port HTTP standar, sekaligus menyediakan antarmuka monitoring **Gatus** pada port yang terpisah.

Elastic Container Service (ECS)

Aplikasi memerlukan sebuah **platform orkestrasi container** untuk mengelola dan men-deploy layanan berbasis container. Anda harus membuat **ECS Cluster** dengan nama **lks-oss-cluster** dan mengaktifkan **AWS Fargate (Serverless)** pada bagian infrastruktur untuk menghilangkan kebutuhan pengelolaan server, sehingga Anda dapat fokus sepenuhnya pada proses deployment dan menjalankan aplikasi container.

Di dalam ECS cluster yang telah dibuat, definisikan sebuah **ECS Service** untuk menjalankan aplikasi **Leantime** dengan konfigurasi berikut:

- **Compute configuration**: Atur **capacity provider strategy** ke **Fargate**, set **base value** ke **0** dan **weight** ke **1**, serta pilih **platform version terbaru**
- **Deployment configuration**: Gunakan **leantime task definition** dengan versi terbaru dan beri nama service **leantime-service**
- **Network configuration**: Gunakan **private subnet** dari VPC yang telah dikonfigurasi, nonaktifkan **public IP assignment**, dan pilih **security group** yang telah dibuat untuk aplikasi guna meningkatkan keamanan
- **Load balancing configuration**: Aktifkan **load balancing** dan pilih **Application Load Balancer** serta **target group** yang telah dikonfigurasi untuk aplikasi Leantime agar task terdistribusi dengan baik
- **Service Auto Scaling configuration**: Aktifkan **auto scaling** dengan jumlah task minimum **1** dan maksimum **5**. Gunakan **Target Scaling** sebagai tipe scaling. Konfigurasikan kebijakan scaling berdasarkan **CPU utilization 75%** dan atur **cooldown scale-out** serta **scale-in** masing-masing **300 detik**

Buat **ECS Service kedua** untuk menjalankan aplikasi monitoring **Gatus** di dalam cluster dengan konfigurasi berikut:

- **Compute configuration**: Atur **capacity provider strategy** ke **Fargate**, set **base value** ke **0** dan **weight** ke **1**, serta pilih **platform version terbaru**
- **Deployment configuration**: Gunakan **Gatus task definition** dengan versi terbaru, beri nama service **gatus-service**, dan atur **desired number of tasks** menjadi **2**

- **Network configuration**: Gunakan **private subnet** dari VPC yang telah dikonfigurasi, nonaktifkan **public IP assignment**, dan pilih **security group** yang telah dibuat untuk aplikasi guna meningkatkan keamanan
- **Load balancing configuration**: Aktifkan **load balancing** dan pilih **Application Load Balancer** serta **target group** yang telah dikonfigurasi untuk aplikasi Gatus agar task terdistribusi dengan baik

Detail Pipeline

GitHub Token

Token digunakan untuk melakukan **autentikasi GitHub Actions**. Buat **Fine-Grained Token hanya** untuk repository Anda. Pada bagian akses repository, atur izin agar dapat **read/write pada Issues** dan **read pada metadata**. Salin dan simpan token ini untuk digunakan di kemudian hari.

Referensi: https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens#creating-a-fine-grained-personal-access-token

GitHub Secret

Tambahkan **secret** ke dalam repository GitHub Anda. Kredensial yang harus ditambahkan ke dalam secret repository adalah sebagai berikut:

- **AWS_ACCESS_KEY_ID**
- **AWS_SECRET_ACCESS_KEY**
- **AWS_SESSION_TOKEN**
- **ALB_ENDPOINT**
- **TOKEN**

GitHub Actions Workflow

Workflow digunakan untuk **mengotomatiskan proses deployment** ke **ECS cluster**. Workflow ini memiliki beberapa **job** yang berjalan secara paralel. Contoh job tersedia di direktori `.github/workflows/` pada file dengan awalan **“example-”**. Oleh karena itu,

perlu melengkapi atau memperbaiki langkah-langkah dan **environment variable workflow** agar job pada workflow dapat berjalan dengan benar. Langkah-langkah yang perlu diisi akan ditandai.

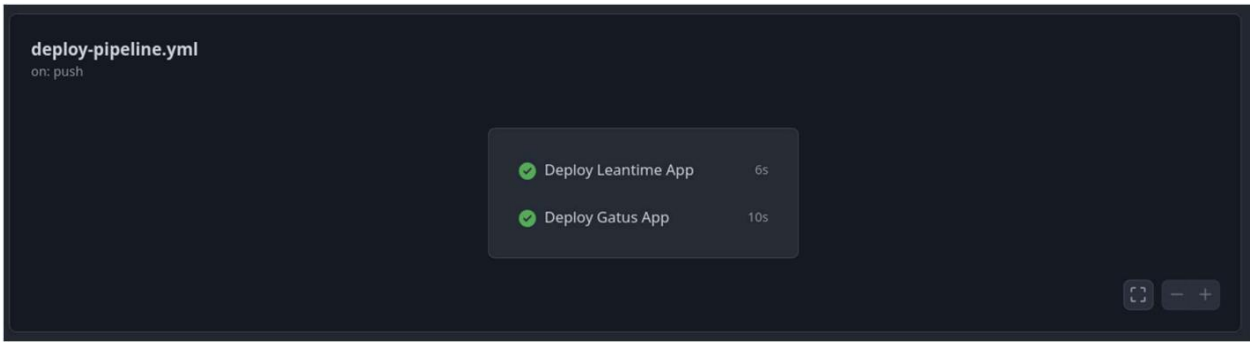
Buat file workflow dengan nama **`lks-deployment-pipeline.yml`**. Atur **trigger event workflow** berdasarkan kode yang di-push ke branch utama pada path tertentu, yaitu:

- `leantime/`
- `gatus/`
- `.github/workflows/lks-deployment-pipeline.yml`

Setiap file memiliki beberapa job dengan rincian sebagai berikut:

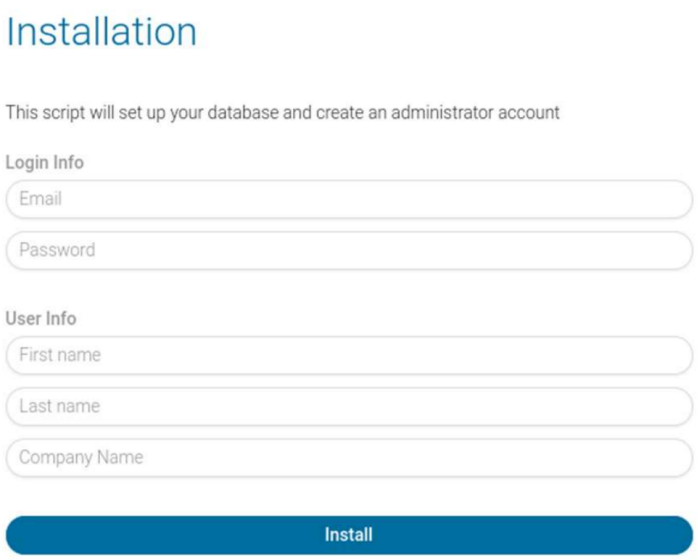
- **Deploy Leantime**: Job ini melakukan render **ECS task definition** untuk aplikasi Leantime dan melakukan deployment ke **ECS service**.
- **Build Gatus**: Job ini melakukan konfigurasi aplikasi, membangun dan memberi tag image, serta mempublikasikannya ke **ECR private registry**. Job ini digabung dengan job **Deploy Gatus**.
- **Deploy Gatus**: Job ini mengonfigurasi dan melakukan render **ECS task definition** untuk aplikasi Gatus, kemudian melakukan deployment ke **ECS service**.

Catatan: Pastikan untuk menghapus seluruh file workflow yang tidak digunakan dan hanya menyisakan file workflow deployment di dalam repository Anda. Contoh tampilan pipeline pada GitHub Actions ditunjukkan sebagai berikut:



Pengujian dan Verifikasi Hasil

Jika proses deployment berhasil, Anda dapat mengakses aplikasi melalui **endpoint Application Load Balancer (ALB)**. Aplikasi **Leantime** akan menampilkan halaman awal untuk melakukan proses pengaturan aplikasi.



Jika sistem mengalami gangguan atau tidak dapat diakses, Anda akan menerima **notifikasi melalui GitHub Issue** di dalam repository.

MODUL 2

Deployment Automation dan CI/CD