

## MODUL 1

### Implementasi Arsitektur pada AWS

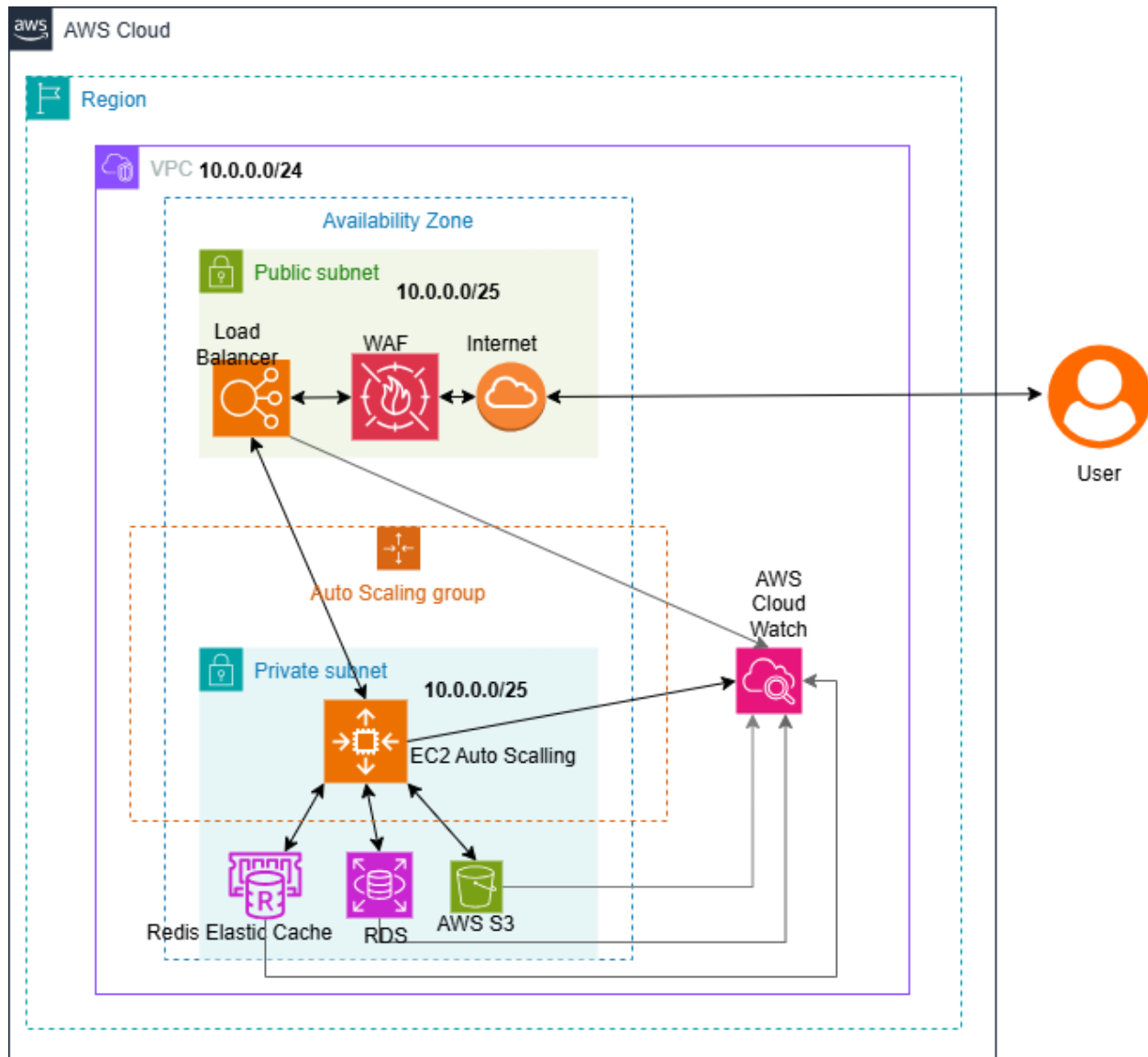
#### Panduan Pengerjaan dan Penilaian

1. Kerjakan sesuai dengan modul.
2. Anda hanya boleh membuka platform AWS dan dokumentasi AWS.
3. Buatlah dokumentasi di word, lalu di ekspor PDF dan diunggah di link <https://tel-u.ac.id/modul1ccsda26>
4. Penilaian berdasarkan keberhasilan implementasi arsitektur dan fungsionalitas.
5. Bagi yang terbukti melakukan Tindakan curang, akan didiskualifikasi.
6. Penilaian Juri tidak dapat diganggu gugat.
7. Modul ini dapat dilihat di <https://bit.ly/modul1CCSDA2026>

## MODUL 1

### Implementasi Arsitektur pada AWS

### Desain dan Deskripsi Arsitektur



Gambar 1 Arsitektur yang Diimplementasikan

Arsitektur tersebut adalah jaringan virtual pribadi di satu region AWS, tempat semua resource (EC2, RDS, Redis, dsb) berada. VPC membuat lingkungan jaringan terisolasi sehingga aturan akses bisa dikontrol penuh dengan subnet, route table, dan security group. Di dalam VPC ada public subnet dan private subnet. Public subnet berisi komponen yang harus terhubung ke internet (Load Balancer, NAT Gateway). Private subnet berisi komponen backend yang tidak langsung diakses dari internet (EC2 aplikasi, Redis, RDS) sehingga lebih aman. ALB berada di public subnet dan bersifat internet-facing sehingga dapat menerima trafik masuk dari luar VPC. Di depan ALB dipasang AWS WAF. WAF memeriksa setiap

## MODUL 1

### Implementasi Arsitektur pada AWS

permintaan HTTP/HTTPS, menerapkan rule seperti proteksi SQL injection/XSS, blok IP yang berbahaya, atau rate-limit, lalu hanya meneruskan request yang lolos ke ALB. Ini menambah lapisan keamanan sebelum trafik mencapai aplikasi. ALB (Elastic Load Balancing) mendistribusikan request ke beberapa instance EC2 di Auto Scaling Group. ALB juga melakukan health check, sehingga hanya instance yang sehat yang akan menerima trafik. EC2 Auto Scaling Group menjalankan beberapa instance EC2 yang berisi aplikasi Laravel. Group ini ditempatkan di private subnet, sehingga tidak memiliki IP publik dan hanya menerima trafik dari ALB. Auto Scaling akan menambah atau mengurangi jumlah instance berdasarkan metric (misalnya CPU atau request) sehingga aplikasi tetap responsif saat beban naik tapi tetap hemat biaya saat beban turun. Setiap EC2 menjalankan web server (Nginx/Apache + PHP) dan aplikasi Laravel. Redis ElastiCache digunakan sebagai penyimpanan session dan cache in-memory. Laravel menyimpan session user ke Redis, sehingga ketika Auto Scaling menambah/mengurangi EC2, session tidak hilang dan user bisa diarahkan ke instance mana saja tanpa harus login ulang. Amazon RDS adalah database terkelola (misalnya MySQL/PostgreSQL) yang menyimpan data persisten seperti user, transaksi, dan konfigurasi aplikasi. RDS berada di private subnet dan hanya dapat diakses dari EC2, bukan langsung dari internet, sehingga keamanan data lebih tinggi. Amazon S3 menyimpan file dan gambar yang di-upload dari aplikasi Laravel (misalnya foto profil, dokumen). EC2 mengunggah file ke bucket S3, sedangkan user mengunduhnya melalui aplikasi atau langsung lewat URL yang dikontrol permission-nya. S3 memberikan penyimpanan yang sangat durabel dan murah. AWS CloudWatch mengumpulkan metric (CPU, memory via agent, network, latency) dan log dari EC2, ALB, RDS, Redis, maupun WAF. Data ini digunakan untuk membuat grafik, alarm, dan notifikasi (misalnya email jika CPU > 80% atau jika banyak request yang diblok oleh WAF). CloudWatch juga menjadi dasar kebijakan Auto Scaling; ketika metric seperti CPU atau jumlah request per target melewati threshold, CloudWatch memicu scaling policy untuk menambah atau mengurangi instance EC2.

### Persiapan Akun AWS

1. Buat/masuk ke akun AWS Free Tier (pastikan kartu kredit sudah diverifikasi).
2. Set region di kanan atas Console

## MODUL 1

### Implementasi Arsitektur pada AWS

## Membuat VPC dan Subnet

1. Buat VPC 10.0.0.0/24. Beri Nama, misalnya (prod-vpc)
2. Buat 2 subnet:
  - Public subnet 10.0.0.0/25 (untuk ALB, NAT). AZ : sama.
  - Private subnet 10.0.0.128/25 (untuk EC2, RDS, Redis). AZ : sama.

Dokumentasi Dasar VPC dan Cara Membuat VPC :

- [https://docs.aws.amazon.com/id\\_id/vpc/latest/userguide/vpc-subnet-basics.html](https://docs.aws.amazon.com/id_id/vpc/latest/userguide/vpc-subnet-basics.html)
- [https://docs.aws.amazon.com/id\\_id/vpc/latest/userguide/create-vpc.html](https://docs.aws.amazon.com/id_id/vpc/latest/userguide/create-vpc.html)

Dokumentasi Subnet dan Cara membuat Subnet :

- [https://docs.aws.amazon.com/id\\_id/vpc/latest/userguide/configure-subnets.html](https://docs.aws.amazon.com/id_id/vpc/latest/userguide/configure-subnets.html)
- [https://docs.aws.amazon.com/id\\_id/vpc/latest/userguide/create-subnets.html](https://docs.aws.amazon.com/id_id/vpc/latest/userguide/create-subnets.html)

## Internet Gateway, Route Table & NAT

### Internet Gateway

1. VPC → Internet Gateways → Create.
2. Set Nama IGW
3. Attach to VPC.

Dokumentasi IGW:

[https://docs.aws.amazon.com/id\\_id/vpc/latest/userguide/VPC\\_Internet\\_Gateway.html](https://docs.aws.amazon.com/id_id/vpc/latest/userguide/VPC_Internet_Gateway.html)

### Route Table

1. **VPC → Route tables → Create route table.**
  - Set Nama, pilih VPC
2. Edit routes:
  - Destination: 0.0.0.0/0, Target: mengarah ke IGW
3. Associate subnet:
  - Arahkan ke public subnet

## MODUL 1

Implementasi Arsitektur pada AWS

Dokumentasi route table:

[https://docs.aws.amazon.com/id\\_id/vpc/latest/userguide/VPC\\_Route\\_Tables.html](https://docs.aws.amazon.com/id_id/vpc/latest/userguide/VPC_Route_Tables.html)

### NAT Gateway

1. Buat **Elastic IP** (EC2 → Elastic IPs → Allocate).
2. **VPC → NAT Gateways → Create NAT gateway:**
  - Atur subnet dan elastic IP
3. Buat **private route table:**
  - Atur nama, VPC, route, arahkan ke private subnet

Dokumentasi NAT Gateway:

[https://docs.aws.amazon.com/id\\_id/vpc/latest/userguide/vpc-nat-gateway.html](https://docs.aws.amazon.com/id_id/vpc/latest/userguide/vpc-nat-gateway.html)

### Security Group & IAM Role Dasar

1. Security Group-ALB
  - a. Inbound: HTTP 80 (dan/atau HTTPS 443) dari 0.0.0.0/0.
  - b. Outbound: all.
2. Security Group EC2
  - a. Inbound: HTTP 80 dari ALB
  - b. SSH 22
  - c. Outbound: all.
3. Security Group -RDS: port DB hanya dari EC2.
4. Security Group -Redis: port 6379 hanya dari EC2
5. Buat **IAM Role untuk EC2** dengan policy minimal S3 dan CloudWatch Agent.

Dokumentasi security group:

[https://docs.aws.amazon.com/id\\_id/vpc/latest/userguide/VPC\\_SecurityGroups.html](https://docs.aws.amazon.com/id_id/vpc/latest/userguide/VPC_SecurityGroups.html)

Dokumentasi IAM role untuk EC2:

[https://docs.aws.amazon.com/id\\_id/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html](https://docs.aws.amazon.com/id_id/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html)

## MODUL 1

### Implementasi Arsitektur pada AWS

#### Create EC2 for AMI

1. Luncurkan 1 EC2 manual dulu (ini yang nanti dijadikan AMI):
  - Pilih OS,
  - pilih instance (bisa pakai t2.micro),
  - Setting Security Group: izinkan HTTP 80 (untuk test) dan SSH 22 dari IP kamu.
  - Subnet: boleh public dulu supaya mudah update & akses.
2. Login ke instance (SSH).
3. Install semua kebutuhan aplikasi (bisa mengikuti readme).
4. Clone dari Repositori : <https://github.com/lkscsda/movieapps.git>  
Ikuti perintah yang ada di repositori
5. Buat service systemd untuk menjalankan php artisan serve otomatis:
6. Uji di browser dengan IP publik EC2: harusnya aplikasi Laravel sudah muncul

Dasar EC2 dan cara launch instance:

[https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2\\_GetStarted.html](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html)

Meluncurkan instance Linux:

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/LaunchingAndUsingInstances.html>

#### RDS (Database Laravel)

1. **RDS Console → Databases → Create.**
2. Pilih Engine sesuai dengan yang digunakan aplikasi
3. Atur Template, nama db, credential, instance class, storage dan connectivity
4. Create database.

Catat endpoint RDS, misalnya laravel-db.xxxxxx.ap-southeast-1.rds.amazonaws.com.

## MODUL 1

Implementasi Arsitektur pada AWS

Dokumentasi RDS untuk instans DB:

[https://docs.aws.amazon.com/id\\_id/AmazonRDS/latest/UserGuide/USER\\_CreateDBInstance.html](https://docs.aws.amazon.com/id_id/AmazonRDS/latest/UserGuide/USER_CreateDBInstance.html)

Konsep keseluruhan RDS:

[https://docs.aws.amazon.com/id\\_id/AmazonRDS/latest/UserGuide/Welcome.html](https://docs.aws.amazon.com/id_id/AmazonRDS/latest/UserGuide/Welcome.html)

## ElastiCache Redis (Session Laravel)

1. **ElastiCache → Redis → Create.**
2. Atur Cluster mode, name, node type, subnet, VPC dan security group
3. Create.

Catat endpoint Redis, misalnya laravel-redis.xxxxxx.ap-southeast-1.cache.amazonaws.com.

Dokumentasi :

Getting started dengan ElastiCache for Redis:

<https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/GettingStarted.html>

Konfigurasi dan parameter Redis:

<https://docs.aws.amazon.com/AmazonElastiCache/latest/redug/RedisConfiguration.html>

## S3 Bucket untuk Upload

1. **S3 Console → Create bucket.**
2. Atur Name, Region, block public access : ON
3. Buat bucket.

Opsional, buat folder uploads/.

Dokumentasi :

## MODUL 1

### Implementasi Arsitektur pada AWS

- Memulai dengan Amazon S3 dan membuat bucket:  
<https://docs.aws.amazon.com/AmazonS3/latest/userguide/GetStartedWithS3.html>
- Mengelola bucket dan objek:  
<https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingBucket.html>

## Membuat AMI

1. Di console EC2 → **Instances** → pilih instance yang sudah Anda buat
2. Actions → *Image and templates* → **Create image**.
3. Beri nama, misalnya ami-laravel-artisan-v1.
4. Biarkan root volume apa adanya (kecuali mau diubah size), klik **Create image**.
5. Tunggu sampai status AMI menjadi **available** di menu **AMIs**.

### Membuat AMI berbasis EBS dari instance EC2:

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami-ebs.html>

## Membuat Template berbasis AMI Custom

1. EC2 → **Launch templates** → **Create launch template**.
2. Atur nama, pilih AMI yang telah dibuat, instance type, IAM role dan security group
3. Atur user data
4. Simpan Launch Template

### Launch template EC2 (menggunakan AMI tertentu):

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-templates.html>

## Auto Scalling Group

1. **EC2** → **Auto Scaling groups** → **Create**.
2. Atur nama, launch template yang telah dibuat
3. Atur Network, pilih VPC, subnet dan AZ

## MODUL 1

### Implementasi Arsitektur pada AWS

4. Load balancing atur nanti
5. Atur Desired capacity: 1, Min capacity: 1, Max capacity: 3.
6. Atur Scaling policies:
  - Target tracking: misalnya CPU 50% .
7. Create.

### Dokumentasi

Apa itu EC2 Auto Scaling:

<https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html>

Membuat Auto Scaling Group dari launch template:

<https://docs.aws.amazon.com/autoscaling/ec2/userguide/create-asg-launch-template.html>

## Setting Load Balancer dengan IP Public

1. **EC2 → Load balancers → Create load balancer → Application Load Balancer.**
2. Atur name, scheme, IP type, VPC, subnet, security group, listener, target group
3. Setelah target group dibuat, hubungkan ke Auto Scaling Group.

### Dokumentasi :

- Pengenalan Application Load Balancer (internet-facing, punya IP publik dinamis):  
<https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html>
- Membuat Application Load Balancer:  
<https://docs.aws.amazon.com/elasticloadbalancing/latest/application/application-load-balancers.html>

## MODUL 1

### Implementasi Arsitektur pada AWS

#### Setting WAF

1. **WAF & Shield → Web ACLs → Create web ACL.**
2. Atur nama, region, resource type, pilih resource
3. Atur Rules:
  - Add **Managed rule groups**:
    - AWSManagedRulesCommonRuleSet (protection dari umum seperti SQLi, XSS).
    - AWSManagedRulesKnownBadInputsRuleSet.
    - AWSManagedRulesAmazonIpReputationList.
  - Tambah **rate-based rule** custom:
    - Name: limit-high-requests.
    - If an IP sends more than 2000 request per 5 minutes, action: block.
4. Default action: allow (karena rules di atas akan block ketika match).
5. Create web ACL.

#### Dokumentasi :

- Pengenalan AWS WAF dan Web ACL:  
<https://docs.aws.amazon.com/waf/latest/developerguide/waf-chapter.html>
- Membuat dan mengonfigurasi Web ACL:  
<https://docs.aws.amazon.com/waf/latest/developerguide/web-acl.html>

#### Integrasi Laravel: RDS, Redis, S3

1. SSH ke EC2. Buka folder aplikasinya.
2. Konfigurasi .env
3. Edit **.env** Laravel agar DB\_HOST mengarah ke endpoint RDS, Redis ke endpoint ElastiCache, FILESYSTEM\_DRIVER=s3 dan bucket S3, APP\_URL bisa dibiarkan atau diisi [http://IP\\_PUBLIK\\_ALB](http://IP_PUBLIK_ALB).

Contoh :

## MODUL 1

### Implementasi Arsitektur pada AWS

```
APP_NAME=Laravel
APP_ENV=production
APP_KEY= # jalankan artisan key:generate
APP_DEBUG=false
APP_URL=http://your-alb-dns.amazonaws.com

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=laravel-db.xxxxxx.ap-southeast-1.rds.amazonaws.com
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=masteruser
DB_PASSWORD=yourpassword

CACHE_DRIVER=redis
SESSION_DRIVER=redis
SESSION_LIFETIME=120

REDIS_HOST=laravel-redis.xxxxxx.ap-southeast-1.cache.amazonaws.com
REDIS_PASSWORD=null
REDIS_PORT=6379

FILESYSTEM_DRIVER=s3

AWS_ACCESS_KEY_ID= # sebaiknya lewat IAM Role, jadi ini bisa
dikosongkan
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=ap-southeast-1
AWS_BUCKET=my-laravel-app-uploads-123
```

4. Jalankan php artisan key:generate, php artisan migrate --force, php artisan config:cache.

Contoh :

```
php artisan key:generate
php artisan config:cache
php artisan migrate --force
```

## MODUL 1

### Implementasi Arsitektur pada AWS

Dokumentasi :

- Menghubungkan aplikasi ke database RDS:  
[https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER\\_ConnectToInstance.html](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ConnectToInstance.html)
- Contoh penggunaan AWS SDK for PHP dengan S3 (untuk upload file dari Laravel):  
<https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/s3-examples.html>
- Konsep koneksi aplikasi ke ElastiCache Redis:  
<https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/GettingStarted.ConnectToCacheNode.html>

## Monitoring CloudWatch

1. EC2, ALB, RDS, ElastiCache, dan S3 otomatis kirim metric dasar (CPU, Network, Latency) ke CloudWatch.
2. Buat CloudWatch Agent
3. Buat alarm CPU untuk Auto Scaling Group, dan alarm WAF untuk banyak serangan.

Contohnya :

- CloudWatch → Alarms → All alarms → Create.
- Metric: EC2 → By Auto Scaling Group → CPUUtilization.
- Pilih asg-laravel-web.
- Condition: > 70% selama 5 menit.
- Action: kirim notifikasi ke SNS (buat topic email).

Dokumentasi CloudWatch metric dan alarm:

[https://docs.aws.amazon.com/id\\_id/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html](https://docs.aws.amazon.com/id_id/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html)

[https://docs.aws.amazon.com/id\\_id/AmazonCloudWatch/latest/monitoring/AlarmThatSendsEmail.html](https://docs.aws.amazon.com/id_id/AmazonCloudWatch/latest/monitoring/AlarmThatSendsEmail.html)

Dokumentasi CloudWatch Agent:

[https://docs.aws.amazon.com/id\\_id/AmazonCloudWatch/latest/monitoring/Install-CloudWatch-Agent.html](https://docs.aws.amazon.com/id_id/AmazonCloudWatch/latest/monitoring/Install-CloudWatch-Agent.html)

## MODUL 1

### Implementasi Arsitektur pada AWS

### Pengujian dengan IP Public

1. Catat IP publik salah satu interface ALB.
2. Dari browser user: akses `http://IP_PUBLIK_ALB`.
3. Pastikan:
  - Halaman Laravel tampil.
  - Login/session tidak hilang (Redis).
  - Upload file tersimpan di S3.
  - Metric di CloudWatch bertambah ketika dites.

Dokumentasi pengujian ALB dan health check:

[https://docs.aws.amazon.com/id\\_id/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html](https://docs.aws.amazon.com/id_id/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html)

ALB :

[https://docs.aws.amazon.com/id\\_id/elasticloadbalancing/latest/application/application-load-balancers.html](https://docs.aws.amazon.com/id_id/elasticloadbalancing/latest/application/application-load-balancers.html)