Day 11: Cloud Deployment – AWS

This day focuses on **end-to-end AWS deployment**, covering compute, database, storage, CDN, serverless, and security. The goal is to understand not just *how* to deploy, but *why* each AWS component is configured the way it is.

---

1. AWS Architecture Overview

Target architecture:

```
User
 ↓
CloudFront (CDN)
 ↓                ↘️
S3 (static)     API Gateway → Lambda
                    ↓
                 EC2 (App)
                    ↓
                 RDS (PostgreSQL)
```

Key principles: - Managed services where possible - Least-privilege security - Cost-aware configuration

---

2. EC2 Deployment (Application Server)

EC2 Instance Setup

- Instance type: `t3.micro` (free-tier eligible)
- OS: Amazon Linux 2023 / Ubuntu 22.04
- Storage: 8–16 GB gp3

Security Group (EC2)

Inbound rules: - SSH (22) → **Your IP only** - HTTP (80) → 0.0.0.0/0 - HTTPS (443) → 0.0.0.0/0

Outbound: - All traffic (default)

Application Deployment (Example)

```
sudo apt update
sudo apt install python3-pip nginx -y
pip install fastapi uvicorn gunicorn
```

Run app via Gunicorn + Uvicorn workers:

```
gunicorn -k uvicorn.workers.UvicornWorker app:app --bind 0.0.0.0:8000
```

✔️ Checklist: EC2 instance running with app deployed

---

3. AWS RDS (Managed PostgreSQL)

RDS Configuration

- Engine: PostgreSQL 15
- Instance class: `db.t3.micro`
- Storage: 20 GB (gp3)
- Public access: ❌ No

Security Group (RDS)

Inbound: - PostgreSQL (5432) → EC2 Security Group only

Connect from EC2

```
psql -h rds-endpoint.amazonaws.com -U app_user -d app_db
```

Run Migrations

```
alembic upgrade head
```

✔️ Checklist: RDS connected and migrations executed

---

4. S3 + CloudFront (Static Assets)

S3 Bucket Setup

- Block public access: ❌ (use CloudFront)

- Versioning: Enabled
- Encryption: SSE-S3

Upload assets:

```
aws s3 sync static/ s3://my-static-bucket
```

CloudFront Distribution

- Origin: S3 bucket
- Viewer protocol policy: Redirect HTTP → HTTPS
- Cache policy: Optimized for static assets

Result:

```
https://d123abcd.cloudfront.net/index.html
```

✔️ Checklist: S3 serving files via CloudFront

---

5. AWS Lambda + API Gateway

Lambda Use Cases

- Lightweight APIs
- Background processing
- Webhooks

Example Lambda Function (Python)

```
def handler(event, context):
    return {
        "statusCode": 200,
        "body": "Hello from Lambda"
    }
```

API Gateway

- Type: HTTP API
- Integration: Lambda
- Auth: None (for demo)

Invoke:

```
curl https://api-id.execute-api.region.amazonaws.com/
```

✔️ Checklist: Lambda triggered via API Gateway

---

6. VPC & Networking

VPC Design

- VPC CIDR: `10.0.0.0/16`
- Public subnet: EC2
- Private subnet: RDS

Routing

- Internet Gateway → Public subnet
- No public route for RDS

---

7. IAM Roles & Least Privilege

EC2 IAM Role

Permissions: - S3 read-only - CloudWatch logs

Lambda IAM Role

Permissions: - Basic execution role - Explicit access only to required services

Principle:

> Never attach `AdministratorAccess` to compute resources.

---

8. Security Groups Summary

Resource	Allowed Inbound
EC2	22 (IP), 80, 443
RDS	5432 from EC2 SG
Lambda	Managed by AWS

✔️ Checklist: Security groups with least privilege

9. Cost Estimation (Monthly – Approx)

Service	Estimated Cost
EC2 t3.micro	$8–10
RDS t3.micro	$15–18
S3 (5 GB)	<$1
CloudFront (low traffic)	<$1
Lambda (low usage)	~$0
Total	**~$25–30/month**

✔️ Checklist: Cost estimation documented

Day 11 Completion Status

- ☑️ EC2 application deployed
- ☑️ RDS connected & migrations run
- ☑️ S3 + CloudFront configured
- ☑️ Lambda + API Gateway working
- ☑️ VPC, IAM, Security Groups hardened
- ☑️ Cost estimation prepared

Key Interview Insight

> Cloud deployment is not about clicking buttons. It is about **security, isolation, scalability, and cost control**.

Next possible steps: - **Day 12: CI/CD Pipelines (GitHub Actions, AWS CodeDeploy)** - Convert Days 6–11 into a **complete production architecture case study** - Add **autoscaling + load balancer** on EC2

Tell me how you want to continue.