

FundifyHub — Cost-first AWS Deployment Guide (Text Only)

Approaches, pros/cons, step-by-step instructions, cost ranges, and a decision flowchart.

Generated: 7/11/2025, 11:14:16 pm

Contents

- Overview
- Approach A — Single EC2 + RDS
- Approach B — ECS Fargate + RDS
- Approach C — Serverless (Lambda + Aurora/RDS Proxy)
- Approach D — Hybrid (ECS web/API + EC2 worker)
- Decision Flowchart
- Runbook checklist

Overview

This guide proposes four deployment approaches with a cost-first mindset. Each section includes tradeoffs, a concrete setup checklist, and rough monthly cost ranges for 100/500/1000 monthly users.

Approach A — Single EC2 + RDS

When to pick

- You want the lowest steady monthly cost and straightforward ops.
- Traffic is moderate and uptime requirements are reasonable (single-AZ).

Pros

- Lowest ongoing cost for always-on workloads.
- Simple debugging (SSH).
- No container orchestration complexity.

Cons

- Single point of failure if single-AZ.
- You maintain OS, Docker, and patching.
- Manual scaling (add another instance or migrate later).

Step-by-step

- Create RDS: PostgreSQL, db.t4g.small (or t3.small), gp3 20GB, single-AZ, private subnet.
- Launch EC2: Ubuntu 22.04 LTS, t4g.medium (or t3.medium), 20GB gp3, SG rules for 80/443 from world, 5432 from EC2 !' RDS.
- Install Docker & Compose. Pull/build images or run docker-compose for frontend, backend, worker, nginx, redis.
- Set environment variables (DATABASE_URL to RDS; secrets via .env or AWS Secrets Manager).
- Run Prisma migrations and seed against RDS.
- Set up TLS (ACM + nginx) and CloudWatch metrics/alarms.

Estimated monthly cost — Approach A

| Scenario | Monthly (low) | Monthly (high) |
|------------------------|---------------|----------------|
| 100 users (~10k req) | \$45 | \$95 |
| 500 users (~50k req) | \$60 | \$150 |
| 1000 users (~100k req) | \$100 | \$250 |

Assumptions:

- EC2 t3/t4g.medium + EBS gp3; RDS db.t3/t4g.small single-AZ; minimal data transfer.
- Redis on EC2 or ElastiCache (t4g.micro) when needed.

Approach B — ECS Fargate + RDS

When to pick

- You prefer managed container orchestration and easier CI/CD.
- Willing to pay a bit more for reduced ops effort.

Pros

- No EC2 to patch; tasks scale horizontally.
- Good ALB integration and blue/green deployments.

Cons

- Higher base cost for always-on services.
- Task sizing requires tuning to control spend.

Step-by-step

- Build/push images to ECR; create RDS as in Approach A.
- Create ECS cluster (Fargate). Define Task Definitions (frontend/api, worker).
- Create ALB + target groups + listeners (HTTPS via ACM).
- Configure service autoscaling and environment secrets.
- Wire CI/CD (GitHub Actions) to push images and update services.

Estimated monthly cost — Approach B

| Scenario | Monthly (low) | Monthly (high) |
|------------|---------------|----------------|
| 100 users | \$80 | \$160 |
| 500 users | \$120 | \$300 |
| 1000 users | \$200 | \$500 |

Assumptions:

- 2–3 small Fargate tasks + RDS single-AZ + ALB + data transfer.

Approach C — Serverless (Lambda + Aurora Serverless or RDS Proxy)

When to pick

- Workload is spiky or low-average; strong need for auto-scaling.
- You want minimal operations at the expense of tuning cold starts and DB connections.

Pros

- Scales to zero for low traffic (lower idle cost).
- Very low ops overhead.

Cons

- Can get pricey for steady traffic; careful with concurrency and DB connections.
- More moving parts (functions, API Gateway/ALB, DB pooling).

Step-by-step

- Package Next.js backend as serverless handlers or container-based Lambdas.
- Use Aurora Serverless v2 or RDS + RDS Proxy. Limit connections and pool aggressively.
- Use SQS for background jobs or separate small Lambdas triggered by SQS.
- Set infra with IaC (SAM/CDK) and deploy via CI/CD.

Estimated monthly cost — Approach C

| Scenario | Monthly (low) | Monthly (high) |
|------------|---------------|----------------|
| 100 users | \$20 | \$80 |
| 500 users | \$60 | \$220 |
| 1000 users | \$120 | \$500 |

Assumptions:

- Lambda invocations + Aurora Srvls v2 (or RDS Proxy) + API Gateway/ALB.

Approach D — Hybrid (ECS for web/API, EC2 for workers)

When to pick

- Web/API fits containers well but workers are long-lived/cheaper on EC2.

Pros

- Reduce Fargate cost for long-running workers.
- Keep managed orchestration for web/API.

Cons

- Two operational surfaces (ECS + EC2).

Step-by-step

- Run frontend/API on ECS Fargate behind an ALB.
- Run worker on small EC2 (t4g.small); connect to the same RDS.
- Share VPC + security groups; centralize logs (CloudWatch).

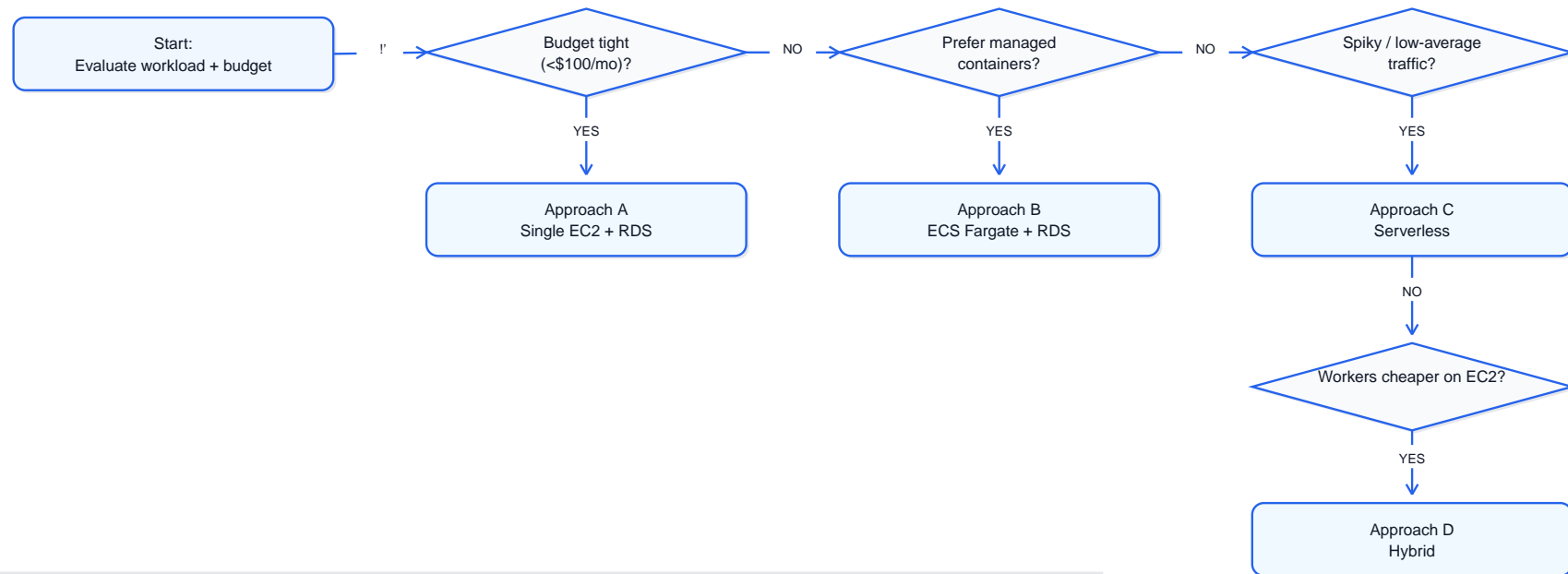
Estimated monthly cost — Approach D

| Scenario | Monthly (low) | Monthly (high) |
|------------|---------------|----------------|
| 100 users | \$80 | \$180 |
| 500 users | \$120 | \$350 |
| 1000 users | \$220 | \$600 |

Assumptions:

- ALB + 1–2 Fargate tasks + 1 EC2 worker + RDS.

Which approach should I choose? (Flowchart)



Runbook checklist

- IAM: minimal roles/policies, rotate keys, use instance/task roles.
- Networking: RDS in private subnets; restrict SG to least privilege.
- Secrets: use AWS Secrets Manager or SSM; avoid committing .env.
- Observability: CloudWatch logs/metrics/alarms, SNS notifications.
- Backups: RDS automated backups; snapshot before destructive changes.