Skills Series

@das-purnendu

01

Your AWS Deployment is Failing (Here's Why)

Most beginners
don't fail
because of code...

They fail because
of these
AWS mistakes

Real stories.
Real mistakes.
Real lessons.

Swipe →

Skills Series

@das-purnendu

02 MISTAKE #1

Hardcoding Secrets in Code ❌

API keys inside your repo = a leak waiting to happen.

What goes wrong:

- Pushed to GitHub → bots scrape it in minutes
- Rotating keys becomes a nightmare
- One leaked key can compromise the whole account

app.py

```
1  import os
2  API_KEY = "AKIAIOSFODNN7EXAMPLE"
3
4  def get_data():
5      headers = {"Authorication":
         f"Bearer {API_KEY}"
6      ...
```

Exposed Secret!

The fix:

AWS Secrets Manager
Secure storage with automatic rotation

SSM Parameter Store
Simple, secure & cost-effective

IAM Roles
For service-to-service auth, no keys needed

Pro Tip: Never store long-lived credentials in code. Use IAM roles wherever possible.

Skills Series

@das-purnendu

03 MISTAKE #2

Wrong IAM Permissions

Too much access → Security risk

Too little access → App breaks

The classic mistake:
slapping "Action": "*" and "Resource": "*"
on a role to “make it work.”

Too Much Access (Dangerous)

```
{
  "Effect": "Allow",
  "Action": "*",
  "Resource": "*"
}
```

Huge security risk!

One bug = full account compromise

Too Little Access (Breaks App)

AccessDeniedException

User: arn:aws:sts::123456789012:
assumed-role/MyRole is not
authorized to perform:
`s3:PutObject` on resource:
`arn:aws:s3:::my-bucket/*`

App fails, and you have no idea why!

The Fix:

Principle of Least Privilege
Give only what’s needed, nothing more.

Use IAM Roles, Not Long-lived Keys
Roles are temporary and more secure.

Test Policies with IAM Policy Simulator
Validate before you deploy.

Pro Tip: Review CloudTrail logs. They show exactly what was allowed or denied – super helpful for debugging permissions.

Skills Series

@das-purnendu

04 MISTAKE #3

No Logging or Monitoring

App fails... you don't know why.
No logs = Blind debugging.

WITHOUT LOGS

```
$ pm2 logs

No logs found

...
```

- You SSH into servers
- You add print() everywhere
- You still don't know the root cause

Result: Hours wasted. Users angry.

WITH LOGGING & MONITORING

CloudWatch Dashboard

Errors (5xx)	Latency (p95)	Requests
12	320 ms	15.6K

Log events

```
INFO   2024-05-10 10:15:21   Request started  /api/orders
INFO   2024-05-10 10:15:21   User authenticated userId=123
ERROR  2024-05-10 10:15:22   Payment failed  reason=INSUFFICIENT_FUNDS
INFO   2024-05-10 10:15:23   Request completed  200  120ms
```

- Know what's happening in real-time
- Get alerted before users complain
- Debug faster, fix faster

Result: Happy users. Peaceful nights.

THE FIX:

CloudWatch Logs
Centralize all your application logs.

CloudWatch Alarms
Alert on errors, latency, 5xx, CPU, etc.

AWS X-Ray
Trace requests across microservices.

Structured Logs
Use JSON logs. Searchable & powerful.

Pro Tip: Set alarms on key metrics (Errors, Latency, CPU, 5xx).
Getting alerted early saves your app (and your reputation!).

Skills Series

@das-purnendu

05 MISTAKE #4

Ignoring Scalability

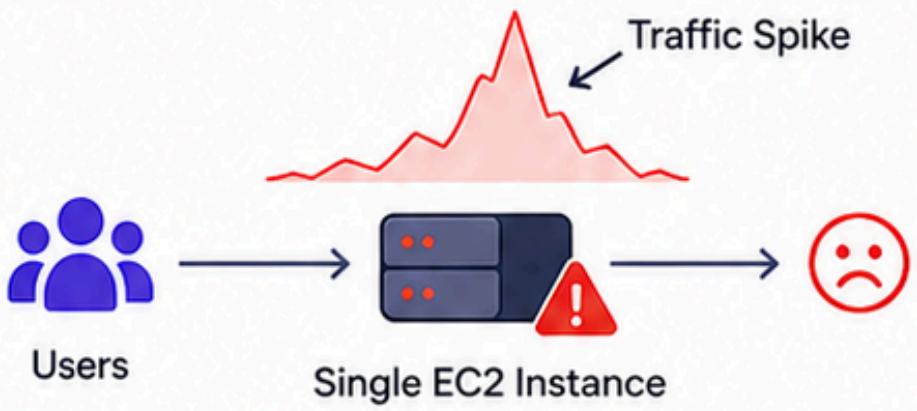
Works locally

Crashes with users

Traffic spike =

WITHOUT SCALING

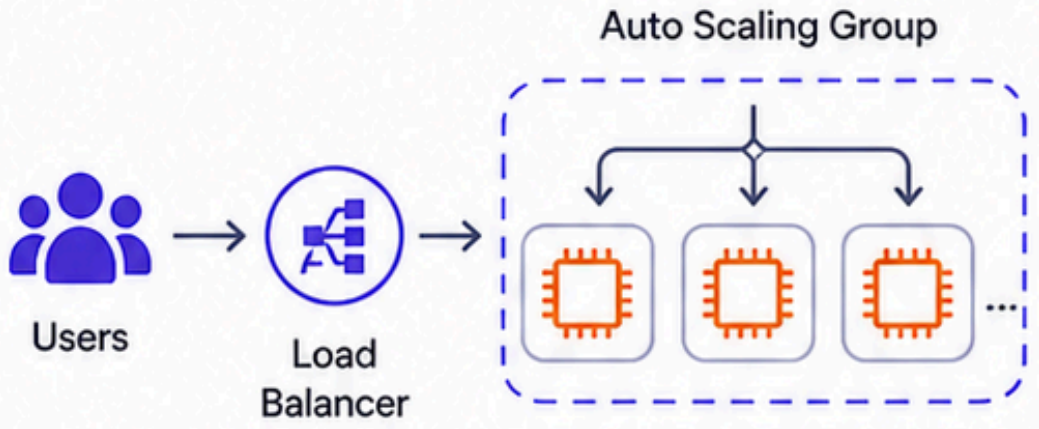
Single server = Single point of failure



- Server overloaded
- High downtime
- Poor user experience

WITH SCALING

Distribute traffic. Scale automatically.



- Handles traffic spikes
- High availability
- Happy users 😀

THE FIX:

Auto Scaling Groups
Automatically adds or removes EC2 instances based on demand.

Application Load Balancer
Distributes traffic across healthy instances.

Design Stateless
Store sessions in ElastiCache or DynamoDB.

Serverless Options
Use Lambda or Fargate for unpredictable traffic.

Pro Tip: Test with realistic load early. Tools like k6, Locust or Artillery help you find limits before your users do.

Skills Series

@das-purnendu

06 BONUS MISTAKE

Ignoring Cost Optimization

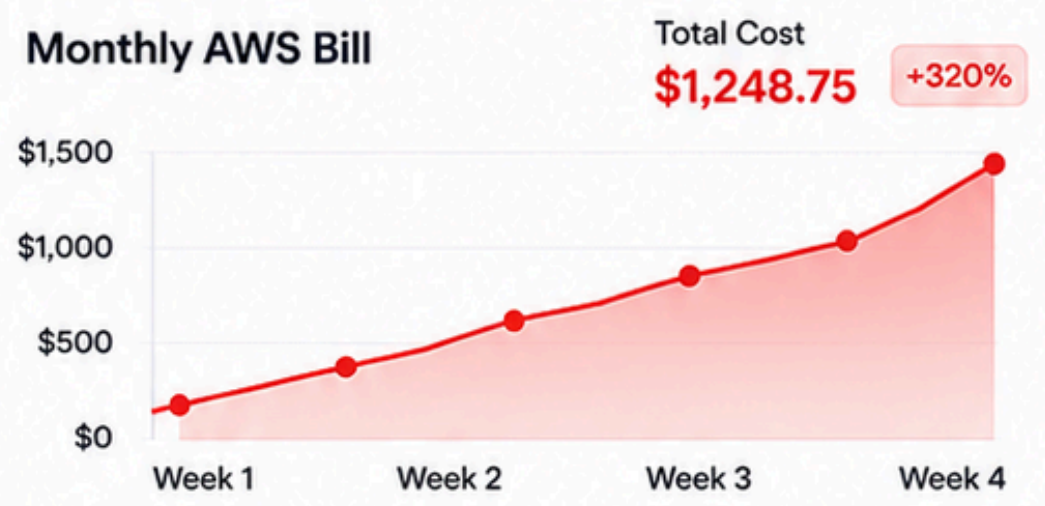
AWS doesn't stop charging because you stopped using it.

Real Story (Happens More Than You Think)

An intern launched an EC2 instance
for a test. Forgot to stop it.
It ran for weeks on a t3.2xlarge.

Result: Unexpected bill & angry email 😬

Your Bill Can Grow Silently



Common Silent Cost Killers

EC2 instances left running
Even weekends cost $$$.

Unattached EBS volumes
Still charged even when not in use.

NAT Gateways in dev
Small traffic, huge charges.

Idle RDS instances
Running but not being used.

Orphaned resources (ELB, IPs, Snapshots)
Small resources add up.

The Fix: Control Cost Before It Controls You

Tag Everything
Use tags like *env*, project, owner to track costs.

Set AWS Budgets
Create budgets and get email alerts.

Billing Alerts
Use CloudWatch billing alarms for early warning.

Review Weekly
Use Cost Explorer. Find & eliminate waste.

Pro Tip: Enable Cost Anomaly Detection. AWS will alert you when spending is higher than usual.

Skills Series

@das-purnendu

07 FINAL SLIDE

🚀 Become Job-Ready in AWS

(Not Just Tutorial-Ready)

Most people learn AWS...
Very few understand
real deployment problems.

That's where you win.

Follow: @das-purnendu — Follow

Comment: AWS ROADMAP — Comment

Save this post — Save

Share with your friends — Share

Like | Comment | Share | Save

Learn. Build. Deploy. Repeat.
Real skills. Real projects. Real jobs. ✨

Purnendu