DevOps Scenario Series – Batch 6: AWS (Beginner to Intermediate Level)

---

1. Scenario: You launched a new EC2 instance using a custom AMI, but it fails the status checks and won't boot properly.

Approach (Reasoning):
When an EC2 instance created from a custom AMI fails to boot, it's often something inside the AMI itself.

- First, use **EC2 Console → Actions → Get System Log** to see if the boot process is failing at a kernel level, missing drivers, or failing cloud-init.
- Check if the **root EBS volume** exists, is correctly attached, and contains a **valid OS**.
- If you can't access logs directly, **detach the root volume**, attach it to another working instance, and manually review boot logs (/var/log/boot.log, /var/log/cloud-init.log).
- If you made kernel-level changes in the AMI, it may not be compatible with the current instance type.

---

2. Scenario: You uploaded files to an S3 bucket, but when trying to access them via CloudFront, you get an "Access Denied" error.

Approach (Reasoning):
This usually points to a permissions mismatch between **CloudFront and S3**.

- If using **Origin Access Identity (OAI)** or **Origin Access Control (OAC)**, make sure it's correctly attached to the CloudFront distribution and has proper access to the S3 bucket.
- Go to the **S3 bucket policy**, and ensure it allows the OAI/OAC to perform s3:GetObject.
- If files were uploaded by another AWS user or service, check **S3 Object Ownership** settings — ideally set to "Bucket owner preferred."
- Lastly, check if the object is public, and confirm CloudFront is not caching an old denied version.

---

3. Scenario: A developer deployed a new version of a Lambda function, but it's still executing the old version during invocation.

Approach (Reasoning):
Lambda has versioning and alias support — that's often where the issue lies.

- Confirm whether the Lambda function is being invoked **directly** or via an **alias** (like prod, v1, etc.).

- If an alias is used, make sure it **points to the new version**. Publishing a new version doesn't automatically update the alias.
- Also, check for **caching** — if the function is invoked via **API Gateway** or **CloudFront**, stale cache might still be returning responses from an older version.
- Look at CloudWatch logs to confirm the version invoked ($LATEST vs numbered version).

---

4. Scenario: You set up an Application Load Balancer (ALB), but it returns 502 Bad Gateway errors.

Approach (Reasoning):
A 502 error from ALB means it reached the backend target, but the response was invalid.

- First, check **target group health status** — if it's showing "unhealthy," the issue likely lies in **application readiness** or **health check path**.
- Make sure your app is **listening on the right port and protocol** (e.g., HTTP vs HTTPS).
- Use curl from within the VPC or from another instance to simulate the ALB request — see if the app responds with valid HTTP headers and status codes.
- If the app is slow to start or crashes under load, ALB may time out. Tweak **health check thresholds** accordingly.

---

5. Scenario: You used AWS CLI to launch an EC2 instance, but it ended up in the wrong subnet.

Approach (Reasoning):
CLI defaults can be misleading — especially when subnet or AZ isn't specified.

- Double-check the command — did you provide the --subnet-id and --availability-zone explicitly?
- If not, AWS chooses defaults — often a **random subnet from the default VPC** in that region.
- Run describe-instances to confirm which subnet and AZ the instance landed in.
- For automation, always **explicitly declare subnet and security groups** to avoid misplacement.

---

6. Scenario: Your ECS Fargate task fails to start because it can't pull the image.

Approach (Reasoning):
This usually ties back to **registry permissions** or incorrect image URIs.

- Make sure the **container image URI** is correctly formatted, especially when using private ECR (e.g., aws_account_id.dkr.ecr.region.amazonaws.com/repo:tag).
- The **execution role** for ECS must have the AmazonEC2ContainerRegistryReadOnly policy to pull from ECR.
- Check if the image is in the **same region** as your Fargate task — if not, enable **cross-region support** or replicate the image.

- Review ECS Events in the console — they often reveal clear error messages.

---

7. Scenario: You keep getting "RequestLimitExceeded" errors while running AWS CLI/API scripts.

Approach (Reasoning):
AWS enforces API rate limits. If you cross them, you'll be throttled.

- Check how often your script or tool is hitting the API. **High-frequency loops or parallel requests** are common culprits.
- Implement **exponential backoff and retries** using AWS SDK or CLI best practices.
- Use **AWS CloudTrail or CloudWatch** to trace which API calls are being throttled.
- If usage is legitimate, go to the **Service Quotas console** and request a **rate limit increase**.

---

8. Scenario: You created a VPC endpoint for S3 but your app in a private subnet can't access it.

Approach (Reasoning):
VPC endpoints require tight configuration between **routing**, **permissions**, and **DNS resolution**.

- Make sure your **private subnet's route table** has a route to the **S3 endpoint** (not the Internet Gateway).
- Check the **VPC endpoint policy** — does it allow access to s3:GetObject on the correct bucket(s)?
- Confirm that your app uses **S3-compliant DNS names** or AWS SDK — raw IPs or non-standard clients won't work with endpoints.
- Also verify that **DNS resolution is enabled** for your VPC if using interface endpoints.

---

9. Scenario: You added an EC2 instance to a target group, but it remains unhealthy in ALB.

Approach (Reasoning):
If the instance is up but marked unhealthy, it's usually an issue with the **health check response**.

- Double-check the **target group's health check path**, protocol, and port — does your app actually respond to /health or whatever you configured?
- From another instance, run curl http://instance_ip:port/health to test.
- If the app is **listening only on localhost**, ALB can't reach it — it needs to be on 0.0.0.0.
- Ensure the **EC2 instance's security group** allows traffic from the ALB on the health check port.

---

10. Scenario: You enabled S3 versioning but still lost data when someone overwrote a file.

Approach (Reasoning):
S3 versioning protects files — but only when **fully enabled and used properly**.

- First, check if **versioning is enabled**, not just "suspended." In suspended mode, overwrites are not protected.
- Use the **S3 console or CLI** to list object versions (aws s3api list-object-versions).
- If files were manually deleted (including all versions), recovery might be impossible.
- To prevent full deletion, set up **S3 Lifecycle rules** — you can archive or transition older versions instead of deleting them outright.
- For extra safety, **enable MFA Delete** for critical buckets.