When it comes to AWS Intermittent failure issues there can be many possible reasons.

First we need to figure out which component is failing. Let's say we have an application that is deployed to AWS lambda that is experiencing this issue.

## Diagnosis steps

**Application diagnosis**

Go to lambda's monitor and cloudwatch logs, we need to look into the application logs to see what is the nature of the failure.

Then check the metrics. We are trying to find out a pattern for this intermittent issues and potentially support our root cause analysis.

In cloudwatch metric check for "invocation" and "error count and success rate" and throttling.

Having some understanding of how the app works would be good. Is the throttling causing timeout hence dropping the requests.

In lambda insights - check memory consumption patterns - is the memory usage too high that causes application to be able to process certain while dropping the rest of the request - hence contributing to intermittent issue.

In lambda insights - check cpu consumption patterns is the cpu usage too high that causes application to be able to process certain while dropping the rest of the request - hence contributing to intermittent issue.

Scale the metrics timeline from 1 - 2 weeks, to see if this has been happening in the past 1 week. If not, then something must have changed recently. We need to figure out what changed on the application side or infrastructure side.

- Use AWS and CloudWatch service lense to reveal service map or service inter-dependencies for the apps, so we have a clear idea of what we're dealing with, if it is a database, a service bus or a caching service.

If the application is working well, we can determine if the intermittent issue is caused by its dependencies. Start by going through the metrics say database - to see if it is accepting connections, check if there’s any downtime of the database or any connection errors.

It is common for applications that establish excessive connections to the database’s max connection count. Those requests that result in a successful connection would be fine but those connections that are not, can cause applications to fail intermittently.

If the application is integrating via an external party, for example, OIDC provider, then we need to see if there's any failure calling their endpoint. Often rate limiting imposed on the external parties can be a factor. Failed calls could contribute to this intermittent failure issue above.

**Network diagnosis**

If the nature of the issue is network related, then we can set up a Network monitor for the VPC that this lambda application is tied to. Here we proceed to configure the source VPC and destination ip. From here, we can get a better understanding if the request is being routed over the network.

If we keep on getting NAT exhaustion issues - where applications are being blocked from initiating connection to external parties, which causes the intermittent issue.

## Resolution

In terms of resolution, it really depends on the diagnosis above. If the issue is due to memory or cpu then we can easily scale this up without downtime.

If it is rate limiting that was imposed then we can request for the rate limiting to be reduced or removed temporarily. This can be a double edge sword, as we don’t really want too many unnecessary requests. We need to figure out why there’s excessive requests generated with the application team and create a hotfix for this issue.

If the issue is due to code, then can we consider rollbacking to the previous version. This can be an expensive operation as we need to know what is the older changeset that we're deploying. Testing is required too to ensure we haven't broken anything.

In this case, if the database max threshold connection is low, we can consider increasing it.

If it is a service dependencies issue, normally with AWS RDS Multi-Az, failover would normally be automatic. We can raise a ticket to AWS for assistance.

If it is a port exhaustion issue then we need to set up a NAT gateway.

## Communication

Depending on the severity of the incident, raise a Production incident and set up a group meeting and pull relevant team members into the call to determine the root cause. This team can often provide an effective workaround, if an application hotfix cannot be released.

In case this is a production issue, we need a person to coordinate and document these events as it happens. This can be useful for post mortem analysis and set the stage for future improvements.