**Monitoring AWS:**

Resources

* <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/monitoring_best_practices.html>
* <https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-rollback-and-redeploy.html>
* <https://d1.awsstatic.com/whitepapers/aws-security-at-scale-logging-in-aws.pdf>
* <https://docs.aws.amazon.com/codebuild/latest/userguide/view-build-details.html>
* <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Query.html>

Key Points

* **System Status Checks**, monitor the AWS systems on which your instance runs. These checks detect underlying problems with your instance that require AWS involvement to repair.
* **Instance Status Checks**, Monitor the software and network configuration of your individual instance. Amazon EC2 checks the health of the instance by sending an address resolution protocol (ARP) request to the ENI. These checks detect problems that require your involvement to repair.
* **Lambda automatically monitors Lambda functions** on your behalf, reporting metrics through Amazon CloudWatch.
* CloudTrail is a web service that records API calls to supported AWS services in your AWS account and delivers a log file to a S3 bucket.
* You can use the AWS CodeBuild console, AWS CLI, or AWS SDKs to view details about builds managed by CodeBuild.
* By default, DynamoDB **Query operation does not return any data on how much read capacity** it consumes. However, you can specify the *ReturnConsumedCapacity* parameter in a Query request to obtain this information.

**Root cause Analysis:**

Resources

* [https://docs.aws.amazon.com/general/latest/gr/api-retries.html](https://docs.aws.amazon.com/lambda/latest/dg/monitoring-functions-access-metrics.html)
* [https://docs.aws.amazon.com/cli/latest/reference/sts/decode-authorization-message.html](https://docs.aws.amazon.com/lambda/latest/dg/monitoring-functions-access-metrics.html)
* <https://aws.amazon.com/blogs/database/monitor-amazon-rds-for-mysql-and-mariadb-logs-with-amazon-cloudwatch/>
* <https://docs.aws.amazon.com/lambda/latest/dg/monitoring-functions-access-metrics.html>
* <https://docs.aws.amazon.com/lambda/latest/dg/dlq.html>
* <https://docs.aws.amazon.com/lambda/latest/dg/using-x-ray.html>

Key Points

* Some AWS operations additionally return an encoded message that can provide details about this authorization failure, use sts *decode-authorization-message --encoded-message <****value****>*
* If you're not using an AWS SDK, you should retry original requests that **receive server (5xx**) or throttling errors. However, **client errors (4xx)** indicate that you need to revise the request to correct the problem before trying again.
* each AWS SDK implements exponential back-off algorithm for better flow control. The idea behind exponential back-off is to use progressively longer waits between retries for consecutive error responses. This shoudlbe used in your application
* **Slow query log** – contains a record of SQL statements that took longer to execute than a set amount of time and that examined more than a defined number of rows. Both thresholds are configurable.
* You can **insert logging statements into your code** to help you validate that your code is working as expected. Lambda automatically integrates with Amazon CloudWatch Logs
* Any Lambda function invoked asynchronously is retried twice before the event is discarded. If the retries fail and you're unsure why, **use Dead Letter Queues (DLQ)** to direct unprocessed events to an Amazon SQS queue or an Amazon SNS topic to analyze the failure.