Developing on AWS (CLI, SDK and IAM Policies)

If you are studying for AWS Developer Associate Exam, this guide will help you with quick revision before the exam. it can use as study notes for your preparation.

Dashboard Other Certification Notes

Developing on AWS (CLI, SDK and IAM Policies)

AWS CLI Dry Runs

- Sometimes, we'd just like to make sure we have the permissions…
- But not actually run the commands!
- Some AWS CLI commands (such as EC2) can become expensive if they succeed, say if we wanted to try to create an EC2 Instance
- Some AWS CLI commands (not all) contain a –dry-run option to simulate API calls

AWS CLI STS Decode Errors

- When you run API calls and they fail, you can get a long error message
- This error message can be decoded using the **STS** command line:
- sts **decode-authorization-message**

AWS EC2 Instance Metadata

- AWS EC2 Instance Metadata is powerful but one of the least known features to developers
- It allows AWS EC2 instances to "learn about themselves" **without using an IAM Role for that purpose.**
- The URL is http://169.254.169.254/latest/meta-data
- You can retrieve the IAM Role name from the metadata, but you CANNOT retrieve the IAM Policy.
- Metadata = Info about the EC2 instance
- Userdata = launch script of the EC2 instance

MFA with CLI

- To use MFA with the CLI, you must create a temporary session
- To do so, you must run the **STS GetSessionToken** API call
- **aws sts get-session-token** –serial-number arn-of-the-mfa-device –token-code code-from-token –duration-seconds 3600

```
{
  "Credentials": {
    "SecretAccessKey": "secret-access-key",
    "SessionToken": "temporary-session-token",
    "Expiration": "expiration-date-time",
    "AccessKeyId": "access-key-id"
  }
}
```

CLI: Command Line Interface

Add user credentials locally using this command:

- $ `aws configure`

If you are using multiple AWS accounts, you can add custom profiles with seperate credentials using this command:

- $ `aws configure --profile {my-other-aws-account}`
- if you you'd like to execute commands on a specific profile:
  - example: `aws s3 ls --profile {my-other-aws-account}`
- if you don't specify the aws profile, the commands will be executed to your **default** profile

AWS CLI on EC2

- IAM roles can be attached to EC2 instances
- IAM roles can come with a policy authorizing exactly what the EC2 instance should be able to do. This is the best practice.
- EC2 Instances can then use these profiles automatically without any additional configurations

CLI STS Decode Errors

- When you run API calls and they fail, you can get a long, encoded error message code
- This error can be decoded using STS
- run the command: `aws sts decode-authorization-message --encoded-message {encoded_message_code}`
- your IAM user must have the correct permissions to use this command by adding the `STS` service to your policy

AWS SDK Overview

- What if you want to perform actions on AWS directly from your applications code ? (without using the CLI).
- You can use an SDK (software development kit) !
- Official SDKs are…
  - Java
  - .NET
  - Node.js
  - PHP
  - Python (named boto3 / botocore)
  - Go
  - Ruby
  - C++
- We have to use the AWS SDK when coding against AWS Services such as DynamoDB
- Fun fact… the AWS CLI uses the Python SDK (boto3)
- The exam expects you to know when you should use an SDK
- We'll practice the AWS SDK when we get to the Lambda functions
- **Good to know:** if you don't specify or configure a default region, then us-east-1 will be chosen by default

AWS Limits (Quotas)

- **API Rate Limits**
  - DescribeInstances API for EC2 has a limit of 100 calls per seconds
  - GetObject on S3 has a limit of 5500 GET per second per prefix • For Intermittent Errors: implement Exponential Backoff
  - For Consistent Errors: request an API throttling limit increase
- **Service Quotas (Service Limits)**
  - Running On-Demand Standard Instances: 1152 vCPU
  - You can request a service limit increase by opening a ticket
  - You can request a service quota increase by using the Service Quotas API

Exponential Backoff (any AWS service)

- If you get ThrottlingException intermittently, use exponential backoff
- Retry mechanism already included in AWS SDK API calls
- Must implement yourself if using the AWS API as-is or in specific cases
  - Must only implement the retries on **5xx server errors** and throttling
  - Do not implement on the 4xx client errors

AWS CLI Credentials Provider Chain

- The CLI will look for credentials in this order

1. Command line options – –region, –output, and –profile
2. Environment variables – AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, and

2. Environment variables – AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, and AWS_SESSION_TOKEN
3. CLI credentials file –aws configure ~/.aws/credentials on Linux / Mac & C:\Users\user.aws\credentials on Windows
4. CLI configuration file – aws configure ~/.aws/config on Linux / macOS & C:\Users\USERNAME.aws\config on Windows
5. Container credentials – for ECS tasks
6. Instance profile credentials – for EC2 Instance Profiles

AWS Credentials Best Practices

- Overall, NEVER EVER STORE AWS CREDENTIALS IN YOUR CODE
- Best practice is for credentials to be inherited from the credentials chain
- If using working within AWS, use IAM Roles
  - => EC2 Instances Roles for EC2 Instances
  - => ECS Roles for ECS tasks
  - => Lambda Roles for Lambda functions
- If working outside of AWS, use environment variables / named profiles