**AWS ‘Developing on AWS’ training course DAY THREE (22/03/2023 – 24/03/2023)**

**Content:**

For details – pls refer to the actual lecture notes/slides

This summary below is just a short & summarized version of the main points in the course.

1. Module 11 (Building a Modern Application):
   1. Monolith application – One single codebase, tightly coupled, while ok for smaller companies, this way of doing things is no longer practical in bigger companies since if there’s an issue, it can be difficult to debug as well, technology stack tends to be restricted/limited (e.g. if original lead decides to use Java for this monolithic codebase, rest of developers have to follow – since ONE codebase only).
   2. Instead of using the “older” approach of monolith application, more modern businesses are moving towards microservices architectures
   3. Microservice architecture/application – Multiple codebases, loosely coupled, easier to debug if things go wrong as we can easily pinpoint and find out which microservice is causing the issue and fix from there (the comparison to monolithic is that in monolithic, need to search through the whole codebase to debug - troublesome). Microservices are more scalable as well, and with each microservice app we can choose and have MORE FREEDOM OF CHOICE to choose our own tech stack.
   4. AWS trainer recommends decoupling your monolithic application and break it up into smaller microservices, in a phased approach (strangler design pattern). Aka transit to microservices INCREMENTALLY…not one whole shot migrate from mono to microservices, as it can cause culture shock and \*may\* cause application breakages/outages (if not migrated properly).
   5. We will also learn the concepts of ‘steps functions’ in AWS.
   6. In summary, AWS ‘Step Functions’ provides a visual interface/service to manage your AWS workflows. You will use the AWS Console to use step functions in AWS.
   7. AWS ‘Step functions’ documentation: <https://aws.amazon.com/step-functions/#:~:text=AWS%20Step%20Functions%20is%20a,machine%20learning%20(ML)%20pipelines>.
2. Module 12 (Granting Access to your Application Users):
   1. Learn about AWS Cognito to explore the authentication/authorization process so as to grant access to users of your application
   2. Amazon Cognito provides an easy to use service for your users to sign up, sign in and manage access to your web/mobile application.
   3. In short AWS Cognito handles the authentication & authorization workflow for your application.
   4. Take note IAM and AWS Cognito are different, even both of them are used for authentication/authorization purposes:
      1. IAM is used for managing access to your AWS resources (Think of it more of Infrastructure Accessibility)
      2. AWS Cognito is used for managing access to user sign up, user sign in, user data management (Think of it more of like Web/Mobile Application or Functionality Accessibility)
   5. AWS Cognito can be used in 2 ways:
      1. User pool (Authentication - Verify the user identity directly in a list of users collected from Amazon Cognito or Social Identity Providers. Think of it like ‘Sign up/Sign In’ 🡪 Users information collected in a pool 🡪 which we will use to verify users sign in, in the future)
      2. Identity pool (Authorization – Does user have the proper permissions to access the AWS resources? The identity pool provides temporary IAM credentials to access the AWS resources. Think of it more of granting temporary, limited-privilege AWS credentials through the aid of external SSO providers -e.g Facebook, and if they verify OK, then we issue the necessary credentials to access our resources)
      3. Both User pool & Identity pool can be used together actually
   6. How AWS Cognito works under the hood (User Pool):
      1. User sign in
      2. AWS Cognito SDKs checks the User pool to authenticate/verify the user if he/she exists
      3. Once user is authenticated/verified, AWS Cognito issues the JWT token that can allow you to access your own server-side resources or AWS API Gateway endpoints.
      4. The JWT token can also be sent to the AWS (Identity pool) to exchange for temporary security AWS/IAM credentials
      5. Those AWS IAM credentials will then be used to check & allow the necessary access to the required AWS resources
      6. So yes, AWS Cognito does use IAM under the hood.
   7. How AWS Cognito works under the hood II (Identity Pool):
      1. User authenticates through external identity providers (e.g. Facebook, Google, SAML SSO)
      2. External Identity provider provides the JWT
      3. App sends JWT to AWS Cognito (Identity Pool) to verify
      4. Once verified and all ok, AWS Cognito issues the temporary AWS credentials tied to the IAM role
      5. User/App (client) sends and uses the AWS credentials to access your AWS resources.
   8. While both approaches seem similar, the main difference between User Pool and Identity Pool is that:
      1. User Pool: AWS STORES the user profiles into a collection/pool which we will verify against. Those profiles could be gathered from a) manual user sign ups in your web app OR b) social identity providers – e.g. Facebook SSO sign in.
      2. Identity Pool: Identity pools DO NOT STORE any user profiles. Authenticate users through external identity provider directly but no storage – e.g. social identity SSO (example – Facebook SSO).
3. Module 13 (Deploying your application):
   1. AWS SAM is an open-source framework that is used to deploy serverless applications.
   2. SAM stands for Serverless Application Model.
   3. AWS SAM generates a CloudFormation template that handles the creation, updating and deletion of your application.
   4. When to use AWS SAM vs CloudFormation?
      1. CloudFormation – for more complex apps
      2. AWS SAM – for more simple apps; more for serverless apps usually
   5. Format of AWS SAM: YAML (Yet another markup language) format
   6. This concept here in this module is essentially teaching how to use AWS SAM:
      1. AWS SAM has a .**yaml template file** which is a CloudFormation template containing information about the application. This is actually very similar to the ecs.yaml (that config file at work) – which is essentially is a CloudFormation template that helps to build & deploy your apps. Or for clearer reference, that template file for deployment in SAM is similar to the ‘serverless.yml’ file I have in my personal twitter coding quotes project.
      2. After making sure you have the template file, we can then install & use the **aws sam cli** to deploy your applications subsequently.
      3. For info on how to use AWS SAM, pls refer to the AWS docs for the various steps: <https://docs.aws.amazon.com/codedeploy/latest/userguide/tutorial-lambda-sam-template.html>