Lecture 9 - Serverless pt.2 - SAM, Step Functions, CI/CD in AWS (1h)

SAM

Dev guide → https://aws.amazon.com/serverless/sam/resources/?sam-blogs.sort-by=item.additionalFields.createdDate&sam-blogs.sort-order=desc

(FAQ is kinda useless)

Features

- A framework to simplify the app development and deployment to the AWS
- Write yaml that converts to CloudFormation (SAM itself is sort of simplified CloudFormation) →
- Allows you to run lambda, api-gw, dynamodb locally (in a containers)
- Requires its own cli to function
  - https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-cli-command-reference.html
- Uses containers to run code locally
- Lots of toolkits for various IDEs available
- Easy to init and deploy (`init → build → transform → package → deploy`)
- Lots of examples →
- For exam:
  - built on CloudFormation (know headers, transform and resources section)
  - commands: `build, package, deploy`
  - SAM policy templates
- In a real life:

    - a good tool to develop serverless applications
- SAR
    - SAM applications repository that is shareable across orgs and accounts (nice to know)
- Workshop https://catalog.us-east-1.prod.workshops.aws/workshops/d21ec850-bab5-4276-af98-a91664f8b161/en-US/

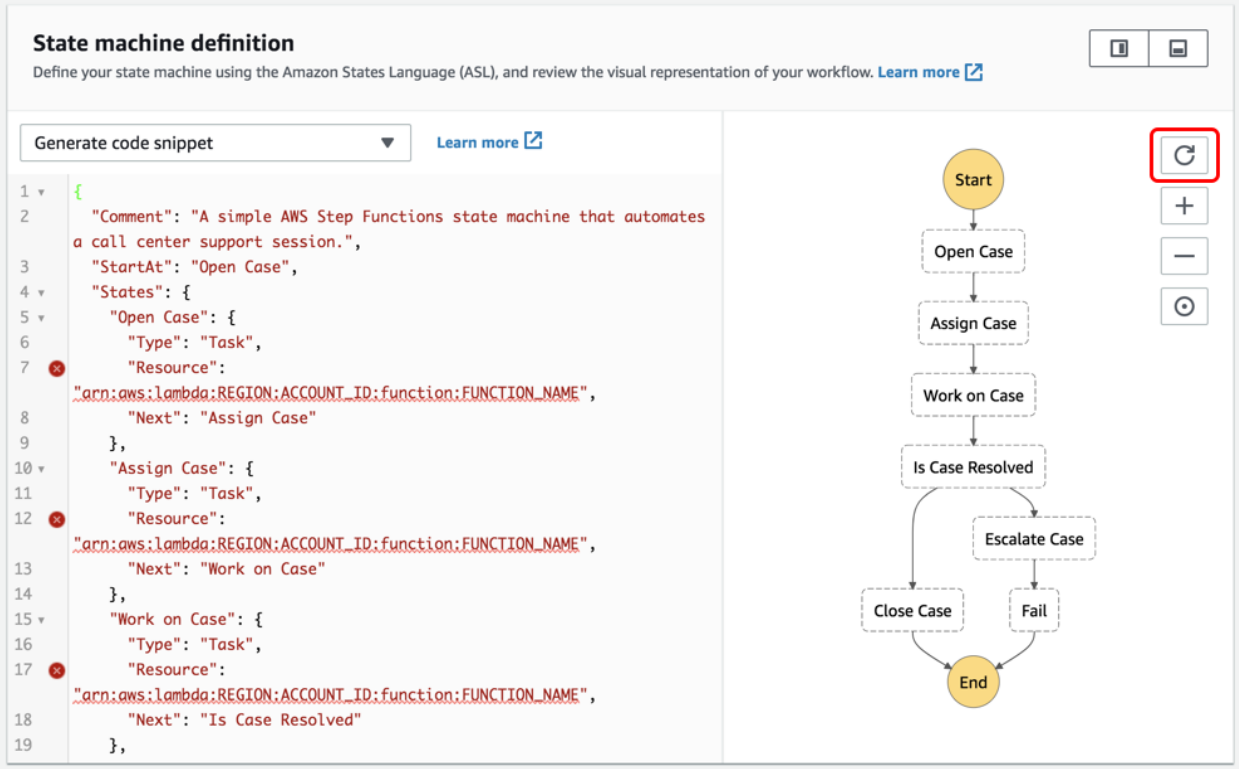
Example

```
# Initializes a new SAM project with required parameters passed as parameters

sam init --runtime python3.7 --dependency-manager pip --app-template hello-world --name sam-app

# Initializes a new SAM project using custom template in a Git/Mercurial repository

# gh being expanded to github url
sam init --location gh:aws-samples/cookiecutter-aws-sam-python

sam init --location git+ssh://git@github.com/aws-samples/cookiecutter-aws-sam-python.git

sam init --location hg+ssh://hg@bitbucket.org/repo/template-name

# Initializes a new SAM project using custom template in a Zipfile

sam init --location /path/to/template.zip

sam init --location https://example.com/path/to/template.zip

# Initializes a new SAM project using cookiecutter template in a local path

sam init --location /path/to/template/folder
```

Step Functions

FAQ → https://aws.amazon.com/step-functions/faqs/

- Orchestrates Lambda Functions, Batch Jobs, ECS tasks or other supported serverless resouress in so-called **Workflows** as **State Machines** → Tutorial: https://aws.amazon.com/getting-started/hands-on/create-a-serverless-workflow-step-functions-lambda/



- Error handling built-in
  - catch
  - retry
- Supports as Task States:
  - Lambda
  - AWS Batch
  - ECS
  - DynamoDB
  - SNS
  - SQS
  - Other StepFunction workflow
- Supported states → https://docs.aws.amazon.com/step-functions/latest/dg/concepts-amazon-states-language.html
- For the certification

    - StepFunctions orchestrates the serverless workflows
    - Uses StateMachine
    - Supports invoking not only Lambda but other services
- Workshop → https://step-functions-workshop.go-aws.com/10_prerequisites.html
- AppSync
    - Workshop → https://catalog.us-east-1.prod.workshops.aws/workshops/67662c95-2007-4281-ae51-5313cd7caa67/en-US/lab0-prereqs
- AWS Amplify
    - Workshop → https://aws.amazon.com/amplify/

Fargate

Guide → https://docs.aws.amazon.com/AmazonECS/latest/userguide/what-is-fargate.html

- Run containerized workflows in a serverless way
- Difference between ECS and Fargate → https://cloudonaut.io/ecs-vs-fargate-whats-the-difference/
- ECS - https://aws.amazon.com/ecs/faqs/

CI/CD in AWS

CodeCommit

FAQ → https://aws.amazon.com/codecommit/faqs/

- managed VCS (git) service in AWS
- IAM access control
- integrates well into other AWS services
- CloudWatch event

Codebuild

FAQ → https://aws.amazon.com/codebuild/faqs/

- managed build service in AWS

- can be used with SAM
- supports builds from any VCS incl. CodeCommit, GitHub or S3
- supports such runtimes → https://docs.aws.amazon.com/codebuild/latest/userguide/runtime-versions.html
- define instructions in **buildspec.yaml**
- output the artifacts to S3, ECR or other artifact storage
- can be run locally

CodeDeploy

FAQ → https://aws.amazon.com/codedeploy/faqs/

- Integrated into SAM
- Capable of *Traffic Shifting* feature
- Pre/post hooks for deployment success validation
- Rollbacks based on CloudWatch alarms
- Supports deployments strategies (like traffic percent per time period → `Canary10Percent10Minutes`)
- define instructions in **appspec.yaml**
- requires agent to run on target machines (if EC2)
- supports deployment groups (like target groups)
- (better explained at Stefaan's course)

CodePipeline

FAQ → https://aws.amazon.com/codepipeline/features/?nc=sn&loc=2

- Combines the CodeCommit, CodeBuild, CodeDeploy into a pipeline
- integrates with the other AWS resources for other stages
- supports multiple deploy options
  - beanstalk
  - cloudformation

    - ecs
    - s3
    - …
- (mostly out of scope but can be asked and good to know)

CodeStar

FAQ → https://aws.amazon.com/codestar/faqs/

- Bootrstrap your app code and dev environment in a browser (Cloud9)
- Automatically integrated with other CodeSuite apps (CodeCommit, CodeBuild, CodeDeploy, CodePipeline)
- Bunch of ready app and infrastructure templates ready (Serverless too)

CodeArtifact

- Artifact management tool in AWS

Q&A session

- topic discussion

Workshops

- SAM
- Step Functions