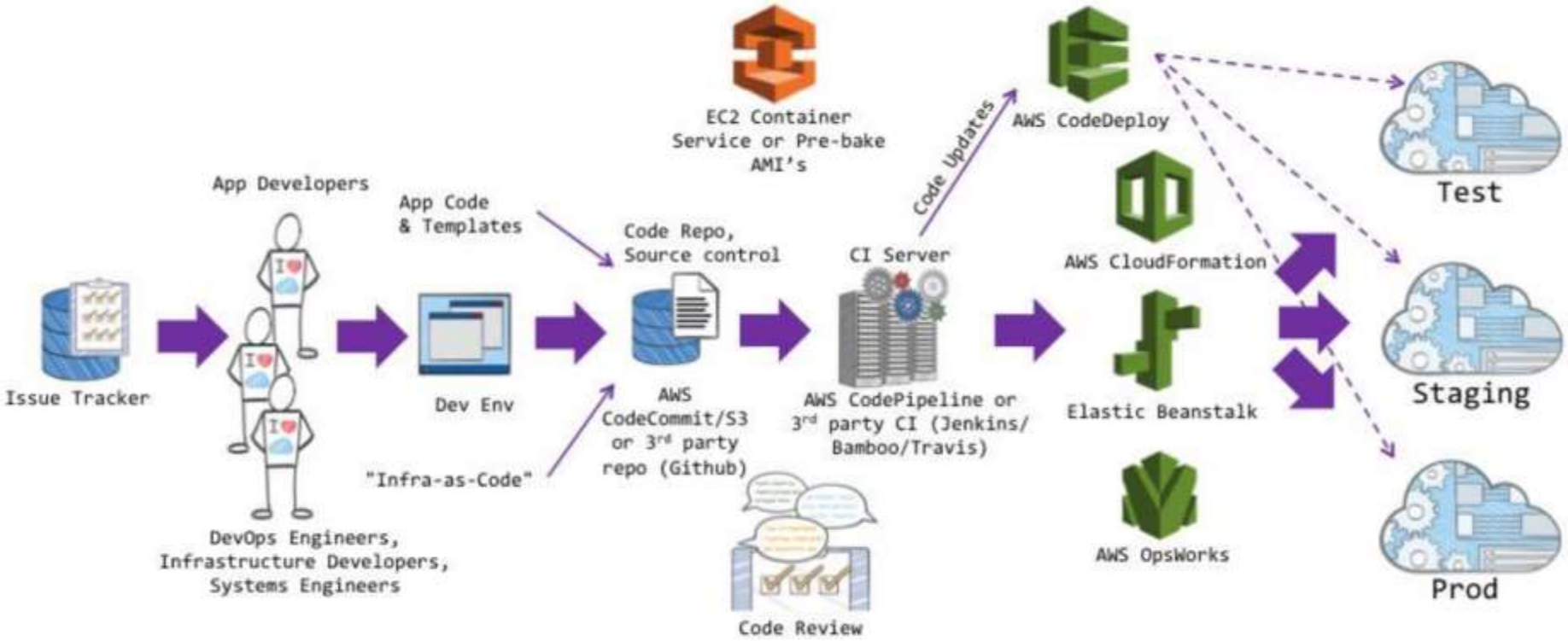
AWS Deployment tools (SAM, Code Pipeline)

AWS Certified Solutions Architect– Professional (SAP-C01)
Study notes - Sep'2019

Overview of Deployment services



Serverless Application Model (SAM)

https://github.com/awslabs/serverless-application-model

A serverless application

- It is a combination of Lambda functions, event sources, and **other resources** that work together to perform tasks.
  - It can include additional resources such as APIs, API Gateway, DataBases, S3 Buckets, DynamoDB tables, and event source mappings.

AWS Serverless Application Model (AWS SAM) is an open-source framework that can be used to simplify building serverless applications on AWS.

- It is a high level abstraction and an extension of Cloudformation

AWS SAM can be used to define (Package, Deploy and Troubleshoot) serverless applications on AWS.

AWS SAM Template specification

- The template specification is in YAML or JSON
- It is used to define the serverless application it represents.
  - It provides a simple and clean syntax to describe the functions, APIs, permissions, configurations, and events that make up a serverless application.
  - An AWS SAM template file is used to operate on a single, deployable, versioned entity that's your serverless application.

AWS SAM command line interface (AWS SAM CLI)

- It is a tool that can be used to build serverless applications that are defined by AWS SAM templates.
  - The CLI provides commands that validates that AWS SAM template files are written according to the specification, It can be used, among other things, to:
    - Define, package, test, and deploy serverless applications
    - Invoke Lambda functions locally (for local testing),
    - Test applications locally before deploying to AWS
    - Step-through debug Lambda functions,

AWS SAM integrates with other AWS services, this provides the following benefits when creati serverless applications with AWS SAM:

- **Single-deployment configuration**
  - AWS SAM makes it easy to organize the serverless application's components and resource operate on a single stack.
  - AWS SAM can be used to share configuration (such as memory and timeouts) between res and deploy all related resources together as a single, versioned entity.
- **AWS SAM is an extension of AWS Cloudformation:**
  - AWS SAM uses the reliable deployment capabilities of AWS CloudFormation.
  - Resources can be defined using AWS CloudFormation in the AWS SAM template

Built-in best practices:

- AWS SAM to define and deploy the serverless application's infrastructure as config.
  - This facilitates the enforcement of best practices, such as code reviews.
  - With a few lines of configuration, safe deployments can be enabled through CodeDeploy,
  - Tracing by using AWS X-Ray can be enabled.

Local debugging and testing

- The AWS SAM CLI allows for locally building, testing, and debugging serverless applications that are defined by AWS SAM templates.

Deep integration with AWS Development tools:

- AWS SAM can be used with a suite of AWS tools to build serverless applications.
  - New applications in the AWS Serverless Application Repository can be discovered
  - For authoring, testing, and debugging AWS SAM–based serverless applications, you can use the AWS Cloud9.
  - To build a deployment pipeline for serverless applications, use CodeBuild, CodeDeploy, and CodePipeline.
  - AWS CodeStar can be used to get started with a project structure, code repository, and a CI/CD pipeline that's automatically configured.

Serverless Application Model (SAM)

AWS Resources that can be declared in an AWS SAM Template

AWS SAM defines the following resources that are specifically designed for serverless applications:

- AWS::Serverless::Api for API Gateway resource
- AWS::Serverless::Function for Lambda function
- AWS::Serverless::LayerVersion
  - This resource type creates a Lambda layer version (LayerVersion) that contains library or runtime code that's needed by a Lambda function
- AWS::Serverless::SimpleTable for DynamoDB tables
  - This resource type provides simple syntax for describing how to create DynamoDB tables
- AWS::Serverless::Application
  - The can be used to embed a serverless application from the AWS Serverless Application Repository or from an Amazon S3 bucket as a nested application.
  - Nested applications are deployed as nested stacks, which can contain multiple other resources.

Using AWS SAM

Initialize the application

- Sample template and app code or create your own.

Test Locally.

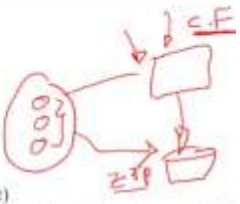
- Test the application locally using sam local invoke and/or sam local start-api.

Package.

- When satisfied with the application components (Lambda, API , DynamoDB…etc)
  - Bundle the components (Lambda functions for ex,) , AWS SAM template, and any dependencies into an AWS CloudFormation deployment package using sam package.
  - The **deployment package** can be used to deploy the application to AWS
- An S3 bucket will be required at this step (new or existing), it will be used to save the packaged code.

Deploy.

- Deploy the application (using the deployment package created) to AWS using sam deploy.
  - Testing the application in the AWS Cloud can be carried out at this stage



AWS SAM can be used to control who can access the API Gateway APIs by enabling authorization within the AWS SAM template.

- AWS SAM supports a few mechanisms for controlling access to your API Gateway APIs:
  - Lambda authorizers.
  - Amazon Cognito user pools.

Serverless Application Model (SAM)

Automating Deployments

AWS SAM can be used with a number of other AWS services to automate the deployment process of serverless application.

- CodeBuild:
  - Use CodeBuild to build, locally test, and package serverless application.
- CodeDeploy (Comes built in with AWS SAM to ensure safe Lambda function deployment):
  - Use CodeDeploy to gradually deploy updates to the serverless applications.
  - If enabled, gradual deployments of the application, new versions/aliases of Lambda functions are automated
- CodePipeline:
  - Use CodePipeline to model, visualize, and automate the steps that are required to release serverless applications

AWS Serverless Application Repository

Publishing Serverless Applications The AWS Serverless Application Repository is a service that hosts serverless applications that are built using AWS SAM. If you want to share serverless applications with others, you can publish them in the AWS Serverless Application Repository. You can also search, browse, and deploy serverless applications that have been published by others.

Working with Logs

AWS SAM CLI has a command called sam logs which fetches logs generated by the Lambda functions from the command line.

AWS SAM Use Case – Process Amazon S3 Events

The application consists of a Lambda function that's invoked by an Amazon S3 object upload event source.

This sample serverless application will:

- Processe object-creation events in Amazon S3.
- For each image that's uploaded to a bucket, Amazon S3 detects the object-created event and invokes a Lambda function.
- The Lambda function invokes Amazon Rekognition to detect text that's in the image.
- It then stores the results returned by Amazon Rekognition in a DynamoDB table.

This use case requires that AWS resources are created and IAM permissions are configured before you can test the Lambda function locally.

- AWS CloudFormation will be leveraged to create the resources and configure the permissions, so they need not be done manually before testing the Lambda function locally.

Serverless Application Model (SAM)

Building Applications with Dependencies

```
# Build a deployment package
$ sam build

# Run the build process inside an AWS Lambda-like Docker container
$ sam build --use-container

# Build and run your functions locally
$ sam build && sam local invoke

# Build and package for deployment
$ sam build && sam package --s3-bucket <bucketname>

# For more options
$ sam build --help
```

Invoking functions locally

```
# Invoking function with event file
$ sam local invoke "Ratings" -e event.json

# Invoking function with event via stdin
$ echo '{"message": "Hey, are you there?" }' | sam local invoke "Ratings"

# For more options
$ sam local invoke --help
```

Run Api gateway locally

```
$ sam local start-api
```

Generating sample event payload

```
#Generates the event from S3 when a new object is created
$ sam local generate-event s3 put

# Generates the event from S3 when an object is deleted
$ sam local generate-event s3 delete
```

Working with logs

- sam logs -n HelloWorldFunction --stack-name mystack
- sam logs -n mystack-HelloWorldFunction-1FJ8PD
- sam logs -n HelloWorldFunction --stack-name mystack --tail
- sam logs -n HelloWorldFunction --stack-name mystack -s '10min ago' -e '2min ago'
- sam logs -n HelloWorldFunction --stack-name mystack --filter "error"

SAM - Gradual Code deployment

https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/automating-updates-to-serverless-apps.html
https://github.com/awslabs/serverless-application-model/blob/master/docs/safe_lambda_deployments.rst

If you use AWS SAM to create your serverless application, it comes built-in with CodeDeploy to help ensure safe Lambda deployments. With just a few lines of configuration, AWS SAM does the following for you:

- Deploys new versions of your Lambda function, and automatically creates aliases that point to the new version.
- Gradually shifts customer traffic to the new version until you're satisfied that it's working as expected, or you roll back the update.
- Defines pre-traffic and post-traffic test functions to verify that the newly deployed code is configured correctly and your application operates as expected.
- Rolls back the deployment if CloudWatch alarms are triggered.

```
Resources:
 MyLambdaFunction:
   Type: AWS::Serverless::Function
   Properties:
     Handler: index.handler
     Runtime: nodejs4.3
     CodeUri: s3://bucket/code.zip

     AutoPublishAlias: live

     DeploymentPreference:
       Type: Canary10Percent10Minutes
       Alarms:
         # A list of alarms that you want to monitor
         - !Ref AliasErrorMetricGreaterThanZeroAlarm
         - !Ref LatestVersionErrorMetricGreaterThanZeroAlarm
       Hooks:
         # Validation Lambda functions that are run before & after tra
         PreTraffic: !Ref PreTrafficLambdaFunction
         PostTraffic: !Ref PostTrafficLambdaFunction
```

AutoPublishAlias:
- Detects when new code is being deployed, based on changes to the function's S3 URI.
- Creates and publishes an updated version of that function with the latest code.
- Creates an alias with a name that you provide (unless an alias already exists), and points to the updated version of the Lambda function.

Deployment Preference Type:
- **Canary**: Traffic is shifted in two increments.
- **Linear**: Traffic is shifted in equal increments with an equal number of minutes between each increment.
- **All-at-once**: All traffic is shifted from the original Lambda function to the updated Lambda function version at once.

Alarms: CloudWatch alarms that are triggered by any errors raised by the deployment. They automatically roll back your deployment.

Hooks: These are pre-traffic and post-traffic test functions that run sanity checks before traffic shifting starts to the new version, and after traffic shifting completes.

Code Deploy

Deployment Configurations on an EC2/On-Premises Compute Platform

Deployment Configuration	Description
CodeDeployDefault.AllAtOnce	
CodeDeployDefault.HalfAtATime	
CodeDeployDefault.OneAtAtTime	

Deployment Configurations on an Amazon ECS Compute Platform

CodeDeployDefault.ECSAllAtOnce - Shifts all traffic to the updated Amazon ECS container at once.

Deployment Configurations on an AWS Lambda Compute Platform:

- **Canary**: Traffic is shifted in two increments. You can choose from predefined canary options that specify the percentage of traffic shifted to your updated Lambda function version in the first increment and the interval, in minutes, before the remaining traffic is shifted in the second increment.
- **Linear**: Traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic shifted in each increment and the number of minutes between each increment.
- **All-at-once**: All traffic is shifted from the original Lambda function to the updated Lambda function version all at once.

Deployment Configuration	Description
CodeDeployDefault.LambdaCanary10Percent5Minutes	Shifts 10 percent of traffic in the first increment. The remaining 90 percent is deployed five minutes later.
CodeDeployDefault.LambdaCanary10Percent10Minutes	Shifts 10 percent of traffic in the first increment. The remaining 90 percent is deployed 10 minutes later.
CodeDeployDefault.LambdaCanary10Percent15Minutes	Shifts 10 percent of traffic in the first increment. The remaining 90 percent is deployed 15 minutes later.
CodeDeployDefault.LambdaCanary10Percent30Minutes	Shifts 10 percent of traffic in the first increment. The remaining 90 percent is deployed 30 minutes later.
CodeDeployDefault.LambdaLinear10PercentEvery1Minute	Shifts 10 percent of traffic every minute until all traffic is shifted.
CodeDeployDefault.LambdaLinear10PercentEvery2Minutes	Shifts 10 percent of traffic every two minutes until all traffic is shifted.
CodeDeployDefault.LambdaLinear10PercentEvery3Minutes	Shifts 10 percent of traffic every three minutes until all traffic is shifted.
CodeDeployDefault.LambdaLinear10PercentEvery10Minutes	Shifts 10 percent of traffic every 10 minutes until all traffic is shifted.
CodeDeployDefault.LambdaAllAtOnce	Shifts all traffic to the updated Lambda functions at once.