IT/Dev CONNECTIONS

Continuous Deployment for Serverless Applications (AWS)

Turner Houghton
Junior Cloud Engineer
Home Technology Services

Delivered by KNect365 TMT an Informa business

What is Continuous Deployment?

- Deployments should be automatically applied to your production environment
- The next step of CI/CD (Continuous Integration, Continuous Delivery)
  - Continuous integration - engineers should merge code frequently to avoid integration problems
  - Continuous delivery - your application should always be in a deployable state
- What characteristics should it have?
  - Instantaneous
  - Transparent
  - Failure resilient

Continuous Deployment - Pros and Cons

Pros

- Ideal for applications that can support frequent version changes
- Can be a background process that "just happens"
- Deploy as often as source control is updated

Cons

- Requires large upfront investment
- Requires constant dedication to automated testing
- Bad for unpredictable/rare deployments

What are Serverless Applications?

- Based on Lambda
  - Run code on demand,
  - invoke other AWS Services
- May use other scalable AWS services:
  - API Gateway
  - Simple Storage Service (S3)
  - DynamoDB
  - Cognito
  - SQS
- Easily defined using CloudFormation/SAM templates

Pipeline Services

- CloudFormation
  - Infrastructure-as-code, YAML/JSON documents
  - Define AWS resources and their properties
  - Create/update stacks
- CodeCommit
  - GIT source control
- CodeBuild
  - Build projects and run tests using buildspec files
  - No managing instances or containers
- CodePipeline
  - Ties everything together
  - Define processes to run in order
  - Monitor each step of deployment

AWS CloudFormation

AWS CodeCommit

AWS CodeBuild

AWS CodePipeline

Simple Pipeline Demo

CodePipeline structure

- For bare minimum functionality, you just need 4 steps:
  - Source
  - Package
  - CloudFormation - CreateChangeSet
  - CloudFormation - ExecuteChangeSet
- If we want an actual CI/CD pipeline, we will need more steps
  - Deploy to multiple stacks
  - Run unit tests
  - Deploy updates to documentation
  - Update static website content
  - Run DB migrations
- You may need more or less steps to fit your application's needs
- Put each phase in it's own CodeBuild project

Deployment Types

- CloudFormation uses rolling deployments
  - Steps through your template, and updates/creates all of your resources
  - If it is unable to create/update a resource, it reverts your entire stack to the latest stable version
  - Use this option with multiple stacks for best results
- SAM has a few extra Lambda-based deployment types
  - SAM is CloudFormation with easier defining of Lambda functions
  - Route between current and previous version of your function
  - Canary (move x%, wait x minutes then move remaining %
  - Linear (move x% every x minutes)
- Any other type of deployment you will have to create yourself

Advanced CodePipeline Demo

Links

- CloudFormation Templates: https://github.com/tqhoughton/cloudformation-templates
- Serverless Demo: https://test.turnerhoughton.com
- LinkedIn: https://linkedin.com/in/turnerhoughton
- Questions? https://reddit.com/r/aws