AWS DevOps Professional Study

Study Notes

These 6 are relevant but already studied for Security Specialty. Review them again if time allows…

~ **KMS** — AWS Key Management Service FAQs

~ **Elastic Beanstalk** — AWS Elastic Beanstalk FAQs

~ **OpsWorks** — AWS OpsWorks – Configuration Management

~ **CloudFormation** — AWS CloudFormation FAQs

~ **Cloudwatch** — Amazon CloudWatch FAQs

~ **AWS Config** — AWS Config FAQs - Amazon Web Services

✔ **AWS CodeStar** — AWS CodeStar FAQs

Guess: Entry point for creating and managing dev teams connectivity to code based projects, provides scaffolding for an entire end-to-end project using CodeCommit + CodeBuild + CodeDeploy via CodePipeline.
After Review: Created a new CodeStar project to figure out how user management works in practice and this also provided a really good end-to-end pipeline because it creates a code commit repo and ties it together with CodeBuild and CodeDeploy via CodePipeline to deploy to a serverless application that you can review for how to set up a deployment pipeline.

✔ **AWS CodeCommit** — AWS CodeCommit FAQs

Guess: AWS managed git repositories, integrates easily with the AWS code* solutions.
After Review: Very limited checks are available by requiring *n* approvers from groups for securing the repo from yolo pull requests. Adding branch protection is done via IAM by denying commits to certain refs in a stringcondition. SSH logins can't be set up from root accounts.

✔ **AWS CodePipeline** AWS CodePipeline FAQs

Guess: CICD orchestrator, orchestrates code building deploying and other integrations e.g. notifications for manual approval.
After Review: Super basic deployment pipeline tech. There's no conditional logic without diverting that to Lambdas or step functions. Spent some time mucking around setting up actions in the pipeline created by CodeStar.

✔ **AWS CodeGuru** Amazon CodeGuru FAQs

Guess: Code analysis.
After Review: Code analysis tooling for Java and Python that gives hotspots, security issue highlights, code best practice highlights and other static analysis. Also tracks performance of code somehow but I doubt that's relevant for the exam because this is limited to two languages.

✔ **AWS CodeArtifact** AWS CodeArtifact FAQs

Guess: ~~Used for storing build artifacts from CodeBuild~~.
After Review: Managed package artifact repositories e.g. nuget/npm/maven etc with optional upstreams so it can be like a pull through cached package provider. To pump packages into this it needs to be done via the native package manager tooling, codebuild can't just yeet packages into your nuget repo.

✔ **AWS CodeDeploy** AWS CodeDeploy FAQs

Guess: A mechanism for deploying code with various strategies e.g. blue green, partial rollouts 50/50 split etc.
After Review: CodeDeploy helps to deploy code to servers by integrating with load balancers and hosts. Host connectivity is via an agent that's installed that does the magic of pulling deployment artifacts, stopping/starting services and running arbitrary scripts. Health checks allow the app back into the production infrastructure and setting up CloudWatch alarms allows you to have a mechanism that rolls back the deployment if the shit hits the fan. Appspec.yml defines the deployment process. Deployment groups are defined by tags or load balancers.

✔ **AWS CodeBuild** AWS CodeBuild FAQs

Guess: CodeBuild is a bit like Teamcity/Gitlab pipelines and Github actions when used to build software.
After Review: Reviewed more about buildspec.yml (Build specification reference for CodeBuild) and created my own codebuild project for a .net core app to learn more about connecting the dots. A buildspec.yml defines the environment that the application is built in and the spec does things like setup, install packages required for the build and then restore build dependencies and build source code. Can publish artifacts to s3 buckets and for codepipeline this uses a special bucket defined by codepipeline

✔ **AWS CDK** [AWS Cloud Development Kit FAQs]

Guess: Pulumi by AWS. Build CloudFormation by code.
After Review: Three primary data types used in the SDK, constructs which define things/resources, stacks which are a collection of constructs contained within a Cloudformation Stack and apps basically contain a collection of stacks/constructs that all form a logical “application”.

✔ **AWS ECS** [Amazon Elastic Container Service FAQs]

Guess: Simple clustering for deploying applications to. Supports services and tasks similar to services and pods in Kubernetes. Load balancer can have the service as a target group then the service routes traffic to the underlying task instances.
After Review: Built a cluster and worked out the difference between Fargate provisioning and manually provisioned nodes. I have no idea why you’d use this over something like EKS if you’re going to go to the effort of provisioning ec2 instances for it to run on, with fargate backing it’s pretty sick though.

✔ **AWS Fargate** [Serverless Compute Engine – AWS Fargate FAQs]

Guess: Managed infrastructure underlying ECS “easy mode” and Fargate backed pods in EKS.
After Review: Basically AWS manages the nodes for your clustering. If you configure EKS to send some pods to Fargate you don’t need to build nodes that match the pod OS requirements you can just magically deploy them and they’ll get allocated to some magic infrastructure in Fargate land and start running. The ease-of-use comes with a cost tradeoff where I estimate Fargate can cost up to >4x as much as self managed nodes.

✔ **AWS Lambda Applications** [AWS Lambda applications]

Guess: A logical bundling of lambda functions that make up an application domain.
After Review: Basically that but it also links some resources like buckets and repositories so it’s easier to understand and manage a collection of Lambda related resources. SAM seems to give you this for free but it can also be set up via the CLI/web console.

Testrun of the AWS Sample Questions

Prior to starting any study I attempt the sample questions to give myself a quick reality-check to find out the areas I'll need to put the most effort into.

Results (70%, 7/10)

1) Failed the question because I thought you could trigger CodePipeline from an SNS topic.
2) Failed half of the multichoice question because I didn't know how to set up branch protection in CodeCommit.
3) Correct.
4) Correct.
5) Failed two thirds of the multichoice question because I misread the question and thought it was asking for a combination of 3 choices that work together but it was asking which options would all solve the problem if they were the only action taken.
6) Correct.
7) Correct.
8) Correct.
9) Correct.
10) Correct.

Gotchas

- CodeCommit doesn't support resource policies https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_identity-vs-resource.html
- CodePipeline and CodeBuild can be triggered by CloudWatch Events.
- Repository changes in CodeCommit can't directly send messages to SNS according to the exam spec but this exists as a feature now in CodeCommit notifications.
- Unsure if Cloudwatch Events will be migrated to Event Bridge, the questions will likely just mention Event Bridge instead of Cloudwatch Events
- Artifacts of codebuild are not encrypted by default, you need to chuck them in an encrypted bucket. This seems out of date codebuild uses KMS wtf I don't get it.

Testrun of the AWS 20 Practice Questions

As a final test of what I've learnt from the study I redo the Sample Questions above to see if I still remember what's going on and if the questions seem easier to comprehend then I do the official 20 practice questions which shows me weak areas I need to do some extra study in.

Results (55%, 11/20) - Rerun (95%, 19/20)

The questions are in random order so I can't number these. 55% is pretty bad but it just gives more areas to study and the practice exam immediately tells you the answer after each question so I use that to do a bit of study into my failures as I fail each question. Pretty soon after failing this practice exam I do it again to see if I can remember the parts I failed and do better the second time around to gain confidence.

1. Failed the entire question because I didn't choose an answer that met the question requirement of having identical build artifacts.
2. Failed the entire question because I wasn't familiar with implementing API Gateway canary deployments.
3. Failed the entire question because I always use CloudFormation instead of SAM but SAM is the best practice solution to all serverless applications.
4. Failed the entire question because I chose to use long term creds instead of STS tokens.
5. Failed the entire question because I didn't know much about ASG termination policies.
6. Failed the entire question because I tried to send logs straight to Kinesis from a CloudWatch agent.
7. Failed the entire question because I tried to invoke a Lambda straight from Trusted Advisor.
8. Failed one third of the multichoice question because I didn't realize DynamoDB streams had to be polled by Lambda.
9. Failed the entire question because I didn't know the default parameters of the EB CLI can't be overridden in .ebextensions files.

Gotchas

- You can use STS credentials on instances instead of access key and secret Use the register-on-premises-instance command.
- Trusted advisor cannot directly invoke lambdas. https://docs.aws.amazon.com/awssupport/latest/user/cloudwatch-events-ta.html
- There are a bunch of instance replacement strategies for ASGs I had no idea about. https://docs.aws.amazon.com/autoscaling/ec2/userguide/ec2-auto-scaling-termination-policies.html?icmpid=docs_ec2as_help_panel
- Cloudwatch agent can't send events to kinesis ya dummy.
- API gateway has built in canary deployments. https://docs.aws.amazon.com/apigateway/latest/developerguide/canary-release.html
- DynamoDB streams need to be polled but the polling is triggered by a notification. https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.Lambda.html
- The elastic beanstalk cli and console page shadow apply shitty defaults that take precedence over .ebextensions configuration values. https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/command-options.html#configuration-options-precedence
- For serverless applications you should use SAM because it's a superset of CloudFormation it can build everything.

Real Exam Results

aws training and certification

AWS Certified DevOps Engineer - Professional

Notice of Exam Results

Candidate: Shaun Lawrie	Exam Date: Aug 19, 2022
Candidate Score: 839	Pass/Fail: PASS

Congratulations! You have successfully completed the AWS Certified DevOps Engineer - Professional and you are now AWS Certified.

AWS Certified Security - Specialty

Breakdown of Exam Results

The information in the table below details the composition of the AWS Certified Security - Specialty and your performance in each of the exam sections. The table includes the classifications of your performance at each **section level**.

This information is designed to provide general feedback concerning your examination performance. The examination is scored using a compensatory scoring model, which means you do not need to "pass" the individual sections. Please keep in mind that each section has a specific weighting on the examination, so some sections have more questions than others. This information is general in nature, highlighting your strengths and weaknesses.

Meets Competencies: Performance at this level demonstrates knowledge, skills, and abilities expected of a passing candidate.

Needs Improvement: Performance at this level does not demonstrate knowledge, skills, and abilities expected of a passing candidate.

		Score Performance	
Section	**% of Scored Items**	**Needs Improvement**	**Meets Competencies**
Domain 1: Incident Response			
Domain 2: Logging and Monitoring			
Domain 3: Infrastructure Security			
Domain 4: Identity and Access Management			
Domain 5: Data Protection			

Disclaimer: AWS Certification exams are designed to make pass/fail decisions based on the total exam score. Section level results are designed to provide direction on areas where a candidate may be weak. Candidates should exercise caution when interpreting the above section level score information as it is less reliable than the total exam score and not intended to guide future test performance.

Gotchas from the Real Exam

- I knew nothing about implementing NAT/internet gateways in various ways which would have been helpful.
- I would have been pretty screwed if I hadn't Implemented an entire Code* solution from code stored in CodeCommit using CodePipelines to integrate with CodeBuild and CodeDeploy. While most knowledge I already had about CICD from Gitlab/Github/Jenkins and TeamCity transfers to the Code* tooling you need to get hands on to understand the tricky bits.
- It would have been better to know more about when tools can integrate directly with SNS vs integrating through Event Hub when sending notifications via SNS.
- I wish I had done some more hands-on stuff with Elastic Beanstalk building more complex applications than just web/data tier.
- SAM is the AWS best practice for serverless applications, I dislike it but need to remember to use it over CloudFormation when given the choice.