- **Vendor: Amazon**
- **Exam Code: AWS Certified DevOps Engineer - Professional**
- **Exam Name: Amazon AWS Certified DevOps Engineer - Professional**
- **Question 81 – End**

QUESTION 81
For AWS Auto Scaling, what is the first transition state an instance enters after leaving steady state when scaling in due to health check failure or decreased load?

A. Terminating
B. Detaching
C. Terminating:Wait
D. EnteringStandby

Answer: A
Explanation:
When Auto Scaling responds to a scale in event, it terminates one or more instances. These instances are detached from the Auto Scaling group and enter the Terminating state.
http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/AutoScalingGroupLifecycle.html

QUESTION 82
You are hired as the new head of operations for a SaaS company. Your CTO has asked you to make debugging any part of your entire operation simpler and as fast as possible. She complains that she has no idea what is going on in the complex, service-oriented architecture, because the developers just log to disk, and it's very hard to find errors in logs on so many services. How can you best meet this requirement and satisfy your CTO?

A. Copy all log files into AWS S3 using a cron job on each instance. Use an S3 Notification Configuration on the <code>PutBucket</code> event and publish events to AWS Lambda. Use the Lambda to analyze logs as soon as they come in and flag issues.
B. Begin using CloudWatch Logs on every service. Stream all Log Groups into S3 objects. Use AWS EMR cluster jobs to perform ad-hoc MapReduce analysis and write new queries when needed.
C. Copy all log files into AWS S3 using a cron job on each instance. Use an S3 Notification Configuration on the <code>PutBucket</code> event and publish events to AWS Kinesis. Use Apache Spark on AWS EMR to perform at-scale stream processing queries on the log chunks and flag issues.
D. Begin using CloudWatch Logs on every service. Stream all Log Groups into an AWS Elasticsearch Service Domain running Kibana 4 and perform log analysis on a search cluster.

Answer: D
Explanation:
The Elasticsearch and Kibana 4 combination is called the ELK Stack, and is designed specifically for real-time, ad-hoc log analysis and aggregation. All other answers introduce extra delay or require pre-defined queries. Amazon Elasticsearch Service is a managed service that makes it easy to deploy, operate, and scale Elasticsearch in the AWS Cloud. Elasticsearch is a popular open-source search and analytics engine for use cases such as log analytics, real-time application monitoring, and click stream analytics.
https://aws.amazon.com/elasticsearch-service/

QUESTION 83
When thinking of AWS Elastic Beanstalk's model, which is true?

A. Applications have many deployments, deployments have many environments.

B. Environments have many applications, applications have many deployments.
C. Applications have many environments, environments have many deployments.
D. Deployments have many environments, environments have many applications.

Answer: C
Explanation:
Applications group logical services. Environments belong to Applications, and typically represent different deployment levels (dev, stage, prod, fo forth). Deployments belong to environments, and are pushes of bundles of code for the environments to run.
http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html

QUESTION 84
You work for a company that automatically tags photographs using artificial neural networks (ANNs), which run on GPUs using C++. You receive millions of images at a time, but only 3 times per day on average. These images are loaded into an AWS S3 bucket you control for you in a batch, and then the customer publishes a JSON-formatted manifest into another S3 bucket you control as well. Each image takes 10 milliseconds to process using a full GPU. Your neural network software requires 5 minutes to bootstrap. Image tags are JSON objects, and you must publish them to an S3 bucket. Which of these is the best system architectures for this system?

A. Create an OpsWorks Stack with two Layers. The first contains lifecycle scripts for launching and bootstrapping an HTTP API on G2 instances for ANN image processing, and the second has an always-on instance which monitors the S3 manifest bucket for new files. When a new file is detected, request instances to boot on the ANN layer. When the instances are booted and the HTTP APIs are up, submit processing requests to individual instances.
B. Make an S3 notification configuration which publishes to AWS Lambda on the manifest bucket. Make the Lambda create a CloudFormation Stack which contains the logic to construct an autoscaling worker tier of EC2 G2 instances with the ANN code on each instance. Create an SQS queue of the images in the manifest. Tear the stack down when the queue is empty.
C. Deploy your ANN code to AWS Lambda as a bundled binary for the C++ extension. Make an S3 notification configuration on the manifest, which publishes to another AWS Lambda running controller code. This controller code publishes all the images in the manifest to AWS Kinesis. Your ANN code Lambda Function uses the Kinesis as an Event Source. The system automatically scales when the stream contains image events.
D. Create an Auto Scaling, Load Balanced Elastic Beanstalk worker tier Application and Environment. Deploy the ANN code to G2 instances in this tier. Set the desired capacity to 1. Make the code periodically check S3 for new manifests. When a new manifest is detected, push all of the images in the manifest into the SQS queue associated with the Elastic Beanstalk worker tier.

Answer: B
Explanation:
The Elastic Beanstalk option is incorrect because it requires a constantly-polling instance, which may break and costs money. The Lambda fleet option is incorrect because AWS Lambda does not support GPU usage. The OpsWorks stack option both requires a constantly-polling instance, and also requires complex timing and capacity planning logic. The CloudFormation option requires no polling, has no always-on instances, and allows arbitrarily fast processing by simply setting the instance count as high as needed.
http://docs.aws.amazon.com/lambda/latest/dg/current-supported-versions.html

QUESTION 85
You are designing a system which needs, at minumum, 8 m4.large instances operating to service traffic. When designing a system for high availability in the us-east-1 region, which has 6 Availability Zones, you company needs to be able to handle death of a full availability zone. How should you distribute the servers, to save as much cost as possible, assuming all of the EC2 nodes are properly linked to an ELB? Your VPC account can utilize us-east-1's AZ's a through f, inclusive.

A. 3 servers in each of AZ's a through d, inclusive.
B. 8 servers in each of AZ's a and b.
C. 2 servers in each of AZ's a through e, inclusive.
D. 4 servers in each of AZ's a through c, inclusive.

Answer: C
Explanation:
You need to design for N+1 redundancy on Availability Zones. ZONE_COUNT = (REQUIRED_INSTANCES / INSTANCE_COUNT_PER_ZONE) + 1. To minimize cost, spread the instances across as many possible zones as you can. By using a though e, you are allocating 5 zones. Using 2 instances, you have 10 total instances. If a single zone fails, you have 4 zones left, with 2 instances each, for a total of 8 instances. By spreading out as much as possible, you have increased cost by only 25% and significantly de-risked an availability zone failure.
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html#concepts-regions-availability-zones

QUESTION 86
You need to create a Route53 record automatically in CloudFormation when not running in production during all launches of a Template. How should you implement this?

A. Use a <code>Parameter</code> for <code>environment</code>, and add a <code>Condition</code> on the Route53 <code>Resource</code> in the template to create the record only when <code>environment</code> is not <code>production</code>.
B. Create two templates, one with the Route53 record value and one with a null value for the record. Use the one without it when deploying to production.
C. Use a <code>Parameter</code> for <code>environment</code>, and add a <code>Condition</code> on the Route53 <code>Resource</code> in the template to create the record with a null string when <code>environment</code> is <code>production</code>.
D. Create two templates, one with the Route53 record and one without it. Use the one without it when deploying to production.

Answer: A
Explanation:
The best way to do this is with one template, and a Condition on the resource. Route53 does not allow null strings for records.
http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/conditions-section-structure.html

QUESTION 87
What is web identity federation?

A. Use of an identity provider like Google or Facebook to become an AWS IAM User.
B. Use of an identity provider like Google or Facebook to exchange for temporary AWS security credentials.
C. Use of AWS IAM User tokens to log in as a Google or Facebook user.
D. Use of AWS STS Tokens to log in as a Google or Facebook user.

Answer: B
Explanation:
... users of your app can sign in using a well-known identity provider (IdP) --such as Login with Amazon, Facebook, Google, or any other OpenID Connect (OIDC)-compatible IdP, receive an authentication token, and then exchange that token for temporary security credentials in AWS that map to an IAM role with permissions to use the resources in your AWS account.
http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_oidc.html

QUESTION 88
You have been asked to de-risk deployments at your company. Specifically, the CEO is concerned about outages that occur because of accidental inconsistencies between Staging and Production, which sometimes cause unexpected behaviors in Production even when Staging tests pass. You already use Docker to get high consistency between Staging and Production for the application environment on your EC2 instances. How do you further de-risk the rest of the execution environment, since in AWS, there are many service components you may use beyond EC2 virtual machines?

A. Develop models of your entire cloud system in CloudFormation. Use this model in Staging and Production to achieve greater parity.
B. Use AWS Config to force the Staging and Production stacks to have configuration parity. Any differences will be detected for you so you are aware of risks.
C. Use AMIs to ensure the whole machine, including the kernel of the virual machines, is consistent, since Docker uses Linux Container (LXC) technology, and we need to make sure the container environment is consistent.
D. Use AWS ECS and Docker clustering. This will make sure that the AMIs and machine sizes are the same across both environments.

Answer: A
Explanation:
Only CloudFormation's JSON Templates allow declarative version control of repeatably deployable models of entire AWS clouds.
https://blogs.aws.amazon.com/application-management/blog/category/Best+practices

QUESTION 89
You are creating a new API for video game scores. Reads are 100 times more common than writes, and the top 1% of scores are read 100 times more frequently than the rest of the scores. What's the best design for this system, using DynamoDB?

A. DynamoDB table with 100x higher read than write throughput, with CloudFront caching.

B. DynamoDB table with roughly equal read and write throughput, with CloudFront caching.
C. DynamoDB table with 100x higher read than write throughput, with ElastiCache caching.
D. DynamoDB table with roughly equal read and write throughput, with ElastiCache caching.

Answer: D
Explanation:
Because the 100x read ratio is mostly driven by a small subset, with caching, only a roughly equal number of reads to writes will miss the cache, since the supermajority will hit the top 1% scores. Knowing we need to set the values roughly equal when using caching, we select AWS ElastiCache, because CloudFront cannot directly cache DynamoDB queries, and ElastiCache is an excellent in-memory cache for database queries, rather than a distributed proxy cache for content delivery. ... One solution would be to cache these reads at the application layer. Caching is a technique that is used in many high-throughput applications, offloading read activity on hot items to the cache rather than to the database. Your application can cache the most popular items in memory, or use a product such as ElastiCache to do the same.
http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GuidelinesForTables.html#GuidelinesForTables.CachePopularItem

QUESTION 90
You were just hired as a DevOps Engineer for a startup. Your startup uses AWS for 100% of their infrastructure. They currently have no automation at all for deployment, and they have had many failures while trying to deploy to production. The company has told you deployment process risk mitigation is the most important thing now, and you have a lot of budget for tools and AWS resources.
Their stack:

```
2-tier API
Data stored in DynamoDB or S3, depending on type
Compute layer is EC2 in Auto Scaling Groups
They use Route53 for DNS pointing to an ELB
An ELB balances load across the EC2 instances
```

The scaling group properly varies between 4 and 12 EC2 servers. Which of the following approaches, given this company's stack and their priorities, best meets the company's needs?

A. Model the stack in AWS Elastic Beanstalk as a single Application with multiple Environments. Use Elastic Beanstalk's Rolling Deploy option to progressively roll out application code changes when promoting across environments.
B. Model the stack in 3 CloudFormation templates: Data layer, compute layer, and networking layer. Write stack deployment and integration testing automation following Blue-Green methodologies.
C. Model the stack in AWS OpsWorks as a single Stack, with 1 compute layer and its associated ELB. Use Chef and App Deployments to automate Rolling Deployment.
D. Model the stack in 1 CloudFormation template, to ensure consistency and dependency graph resolution. Write deployment and integration testing automation following Rolling Deployment methodologies.

Answer: ……