AWS Certified Solution Architect – Important Blogs

Contents

General

All Things distributed – amazon dynamo DB

https://www.allthingsdistributed.com/2007/10/amazons_dynamo.html

Security

Amazon GuardDuty continuous security monitoring threat detection

https://aws.amazon.com/blogs/aws/amazon-guardduty-continuous-security-monitoring-threat-detection/

GuardDuty protect the account by analysing CloudTrail, VPC and DNS logs against know any abonamalties – once any abnormities are identified it reports on its analysis report and can also trigger CloudWatch Event which can be used to trigger a lambda function to remediate the situation. Also, it can send notification to other third-party tool for creating workflow instances. Service is free for a month than after it charge based on per 100,000 logs entry or the size of the log.

Using AWS PrivateLink Integrations to Access SaaS Solutions from APN Partners

https://aws.amazon.com/blogs/apn/using-aws-privatelink-integrations-to-access-saas-solutions-from-apn-partners/

AWS PrivateLink simplifies the security of data shared with cloud-based applications by eliminating the exposure of data to the public Internet. AWS PrivateLink provides private connectivity between VPCs, AWS services, and on-premises applications, securely on the Amazon network. AWS PrivateLink makes it easy to connect services across different accounts and VPCs to significantly simplify the network architecture.

Compute

New Amazon EC2 Feature: Bring Your Own Keypair

https://aws.amazon.com/blogs/aws/new-amazon-ec2-feature-bring-your-own-keypair/

One can import the public key for use with EC2 instance, this is possible to facilitate

1. **Trust** – By importing your own keypair you can ensure that you have complete control over your keys.
2. **Security** -You can be confident that your private key has never been transmitted over the wire.
3. **Management of Multiple Regions** – You can use the same public key across multiple AWS Regions.

New – Use AWS PrivateLink to Access AWS Lambda Over Private AWS Network

https://aws.amazon.com/blogs/aws/new-use-aws-privatelink-to-access-aws-lambda-over-private-aws-network/

Once this feature is enabled privatelink routes all the request through AWS private network. Alternatively DNS name can also be enabled for the lambda function that are made accessible through AWS private network. Once the privatelink route is enabled the lambda function cannot be invoked from the public subnet through the internet gateway, all request needs to be communicated through VPC endpoints in Private subnet.

Automatically updating security groups with lambda function

https://aws.amazon.com/blogs/compute/automating-security-group-updates-with-aws-lambda/

When autoscaling in triggered, it transmits "lifecycle hook" event, which can be used for triggering a lambda function which can list out all the public IP address of that EC2 instance and then update the security group to add the new Ips to allow it passthrough the security group.

Best Practice organizing large serverless applications

https://aws.amazon.com/it/blogs/compute/best-practices-for-organizing-larger-serverless-applications/

- Organized code into multiple repositories – not too small codebase that are difficult to share and not too large codebase that kills reusability. Use SAM to create a deployment pipeline for each of the organized code repositories.
- User AWS service instead of code libraries – for example instead of routing code written in Flask or node.js use API gateway.
- Use separate organization/AWS account for each developers or group of developers, separated from the production environment. Implement CI/CD for deployment of code from/to multiple account.
- Implement CI/CD to manage future releases.

Storage

DynamoDB with Elastic MapReduce

https://aws.amazon.com/articles/using-dynamodb-with-amazon-elastic-mapreduce/

DynamoDB on demand no capacity planning and Pay-per-Request

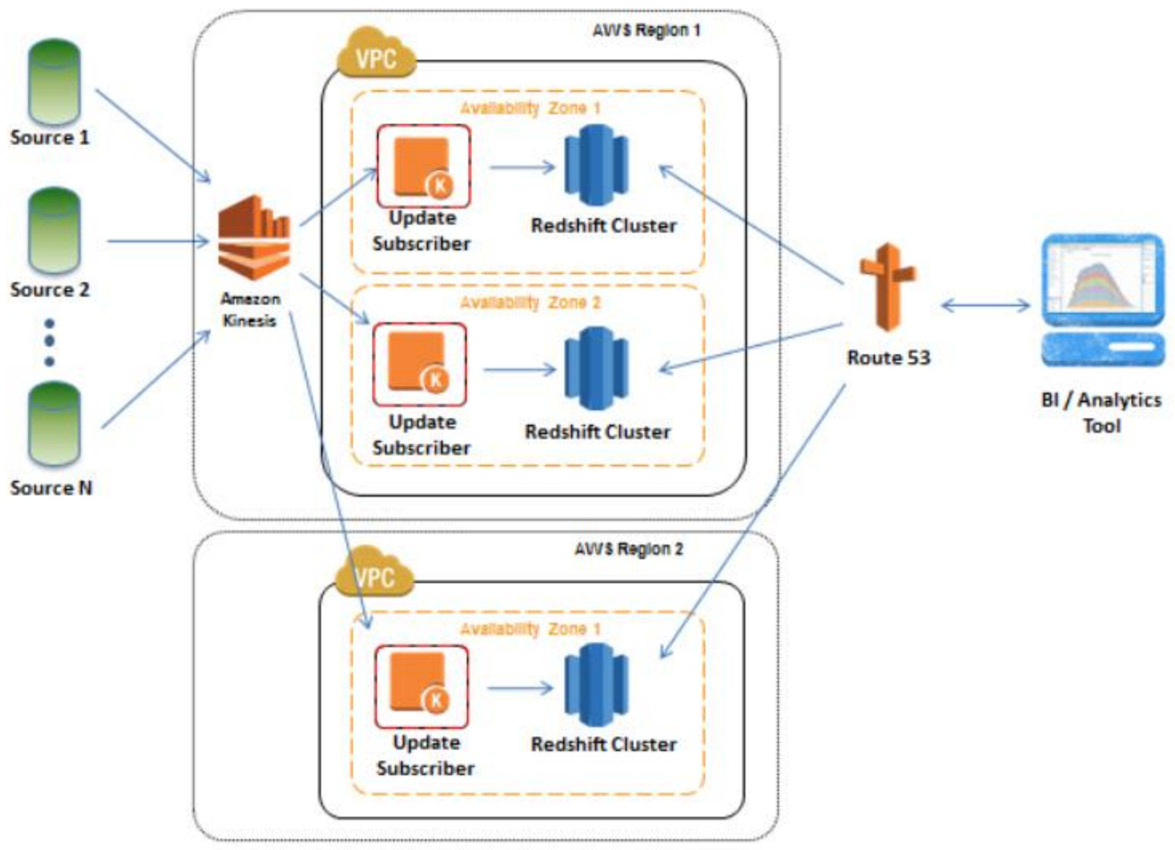
https://aws.amazon.com/blogs/aws/amazon-dynamodb-on-demand-no-capacity-planning-and-pay-per-request-pricing/

EFS Encryption feature

https://aws.amazon.com/blogs/aws/new-encryption-of-data-in-transit-for-amazon-efs/

Multi AZ Multi Region amazon Redshift cluster

https://aws.amazon.com/blogs/big-data/building-multi-az-or-multi-region-amazon-redshift-clusters/



Managing Backup at scale in your AWS Organization using AWS Backup

https://aws.amazon.com/blogs/storage/managing-backups-at-scale-in-your-aws-organizations-using-aws-backup/

IAM

How to access AWS management console using on premises Microsoft AD

https://aws.amazon.com/blogs/security/how-to-access-the-aws-management-console-using-aws-microsoft-ad-and-your-on-premises-credentials/

Demystifying EC2 Resource-Level Permissions

https://aws.amazon.com/blogs/security/demystifying-ec2-resource-level-permissions/

The following policy allow principle to

1. List (describe) all instance.
2. Provide access to - instance, key-pair, Security-Group, Volume, ami-, network-instance, subnet.
3. Provide access to – terminate instance, stop-instance, Start-instance,

One need to use three separates statements within a single policy as not all functions allow resource level access.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TheseActionsDontSupportResourceLevelPermissions",
            "Effect": "Allow",
            "Action": ["ec2:Describe*"],
            "Resource": "*"
        },
        {
            "Sid": "ThisActionSupportsResourceLevelPermissions",
            "Effect": "Allow",
            "Action": ["ec2:RunInstances"],
            "Resource": [
                "arn:aws:ec2:us-east-1:accountid:instance/*",
                "arn:aws:ec2:us-east-1:accountid:key-pair/*",
                "arn:aws:ec2:us-east-1:accountid:security-group/*",
                "arn:aws:ec2:us-east-1:accountid:volume/*",
                "arn:aws:ec2:us-east-1::image/ami-*",
               "arn:aws:ec2:us-east-1:accountid:network-interface/*",
                "arn:aws:ec2:us-east-1:accountid:subnet/*"
               ]
        },
        {
            "Sid": "TheseActionsSupportResourceLevelPermissions",
            "Effect": "Allow",
            "Action": [
                  "ec2:TerminateInstances",
                  "ec2:StopInstances",
                  "ec2:StartInstances"],
            "Resource": "arn:aws:ec2:us-east-1:accountid:instance/*"
        }
    ]
}
```

Writing IAM Policies: Grant Access to User-Specific Folders in an Amazon S3 Bucket

https://aws.amazon.com/blogs/security/writing-iam-policies-grant-access-to-user-specific-folders-in-an-amazon-s3-bucket/

IAM Policy to restrict users' access to their home folder within S3 bucket. S3 doesn't store object in hierarchical structure, instead it stores object as key-value pair however in S3 AWS management console emulates folder like hierarchical structure which makes it possible to apply restriction for user's specific folder access. This is can be archive through IAM policies. Below is a sample IAM policy to restrict user level folder access.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGroupToSeeBucketListInTheConsole",
      "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::*"]
    },
    {
      "Sid": "AllowRootAndHomeListingOfCompanyBucket",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::my-company"],
      "Condition":{"StringEquals":{"s3:prefix":["","home/"],"s3:delimiter":["/"]}}
    },
    {
      "Sid": "AllowListingOfUserFolder",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::my-company"],
      "Condition":{"StringLike":{"s3:prefix":
                    [
                         "home/${aws:username}/*",
                         "home/${aws:username}"
                    ]
                 }
        }
    },
    {
       "Sid": "AllowAllS3ActionsInUserFolder",
       "Action":["s3:*"],
       "Effect":"Allow",
       "Resource": ["arn:aws:s3:::my-company/home/${aws:username}/*"]
    }
  ]
}
```

`"s3:delimiter":["/"]` is not needed for AWS Console however this is advisable to add an additional conditions in case user prefer to access the folder structure through AWS CLI. The delimiter is a character, such as a slash (/), that identifies the folder that an object is in. The delimiter is useful when you want to list objects as if they were in a file system.

Use policy variable to make the policy generic to make it applicable to all user.

AWS Organization

Policy Base management for multiple aws accounts.

https://aws.amazon.com/blogs/aws/aws-organizations-policy-based-management-for-multiple-aws-accounts/

AWS Organization – Best Practice

httsps://aws.amazon.com/blogs/mt/best-practices-for-organizational-units-with-aws-organizations/

CDN

Streaming Videos to mobile application using amazon cloud front

https://aws.amazon.com/blogs/mobile/streaming-videos-to-mobile-app-users-via-amazon-cloudfront-cdn/

Amazon CloudFormation Content Streaming

https://aws.amazon.com/cloudfront/streaming/

Amazon CloudFront Support for Custom Origins

https://aws.amazon.com/blogs/aws/amazon-cloudfront-support-for-custom-origins/

Amazon CloudFront – Support for Dynamic Content

https://aws.amazon.com/blogs/aws/amazon-cloudfront-support-for-dynamic-content/

From now on CloudFront supports dynamic content caching also. (Latest updated on May, 2020). The following are the new features added to enhance the dynamic content caching

1. Persistent TCP Connections
2. Support for Multiple Origins
3. Support for Query Strings
4. Variable Time-To-Live (TTL) (Updated 5/20) link

Origin Configuration	Minimum TTL = 0 Seconds	Minimum TTL > 0 Seconds
The origin adds a Cache-Control max-age directive to objects	CloudFront caching CloudFront caches objects for the lesser of the value of the Cache-Control max-age directive or the value of the CloudFront maximum TTL. Browser caching Browsers cache objects for the value of the Cache-Control max-age directive.	CloudFront caching CloudFront caching depends on the values of the CloudFront minimum TTL and maximum TTL and the Cache-Control max-age directive: Minimum TTL < max-age < maximum TTL CloudFront caches objects for the value of the Cache-Control max-age directive. max-age < minimum TTL CloudFront caches objects for the value of the CloudFront minimum TTL. max-age > maximum TTL CloudFront caches objects for the value of the CloudFront maximum TTL. Browser caching

Origin Configuration	Minimum TTL = 0 Seconds	Minimum TTL > 0 Seconds
		Browsers cache objects for the value of the Cache-Control max-age directive.
The origin does not add a Cache-Control max-age directive to objects	CloudFront caching CloudFront caches objects for the value of the CloudFront default TTL. Browser caching Depends on the browser.	CloudFront caching CloudFront caches objects for the greater of the value of the CloudFront minimum TTL or default TTL. Browser caching Depends on the browser.
The origin adds Cache-Control max-age and Cache-Control s-maxage directives to objects	CloudFront caching CloudFront caches objects for the lesser of the value of the Cache-Control s-maxage directive or the value of the CloudFront maximum TTL. Browser caching Browsers cache objects for the value of the Cache-Control max-age directive.	CloudFront caching CloudFront caching depends on the values of the CloudFront minimum TTL and maximum TTL and the Cache-Control s-maxage directive: Minimum TTL < s-maxage < maximum TTL CloudFront caches objects for the value of the Cache-Control s-maxage directive. s-maxage < minimum TTL CloudFront caches objects for the value of the CloudFront minimum TTL. s-maxage > maximum TTL CloudFront caches objects for the value of the CloudFront maximum TTL. Browser caching Browsers cache objects for the value of the Cache-Control max-age directive.
The origin adds an Expires header to objects	CloudFront caching CloudFront caches objects until the date in the Expires header or for the value of the CloudFront maximum TTL, whichever is sooner. Browser caching Browsers cache objects until the date in the Expires header.	CloudFront caching CloudFront caching depends on the values of the CloudFront minimum TTL and maximum TTL and the Expires header: Minimum TTL < Expires < maximum TTL CloudFront caches objects until the date and time in the Expires header. Expires < minimum TTL

Origin Configuration	Minimum TTL = 0 Seconds	Minimum TTL > 0 Seconds
		CloudFront caches objects for the value of the CloudFront minimum TTL. Expires > maximum TTL CloudFront caches objects for the value of the CloudFront maximum TTL. Browser caching Browsers cache objects until the date and time in the Expires header.
Origin adds Cache-Control: no-cache, no-store, and/or private directives to objects	CloudFront and browsers respect the headers. IF there is no-cache is set , and when a CloudFront edge location receives a request for an object and either the object isn't currently in the cache or the object has expired, CloudFront immediately sends the request to your origin. If there's a traffic spike—if additional requests for the same object arrive at the edge location before your origin responds to the first request—CloudFront pauses briefly before forwarding additional requests for the object to your origin. Typically, the response to the first request will arrive at the CloudFront edge location before the response to subsequent requests. This brief pause helps to reduce unnecessary load on your origin server. If additional requests are not identical because, for example, you configured CloudFront to cache based on request headers or cookies, CloudFront forwards all of the unique requests to your origin.	CloudFront caching CloudFront caches objects for the value of the CloudFront minimum TTL. Browser caching Browsers respect the headers.

5. Large TCP Window
6. API and Management Console Support

Monitoring

Central logging in multi account environment

https://aws.amazon.com/blogs/architecture/central-logging-in-multi-account-environments/

Cross-Account Cross-Region Dashboards with Amazon CloudWatch

https://aws.amazon.com/blogs/aws/cross-account-cross-region-dashboards-with-amazon-cloudwatch/

Cloud Formation

Use CloudFormation StackSet to provision resource across multiple aws account and region

https://aws.amazon.com/blogs/aws/use-cloudformation-stacksets-to-provision-resources-across-multiple-aws-accounts-and-regions/

Cloud Formation Service Role

https://aws.nz/best-practice/cloudformation-service-roles/

This blog explains how one can create service role for cloudFormation which can be leverage by the developers with minimum access (no specific role to create aws resource) to create aws resources using cloudFormation template.

OpsWorks

Autoscaling aws opsworks instances

https://aws.amazon.com/blogs/devops/auto-scaling-aws-opsworks-instances/

CI/DD

Automating safe hands-off development

https://aws.amazon.com/builders-library/automating-safe-hands-off-deployments/

How to build a CI/CD pipeline for container vulnerability scanning with Trivy and AWS Security Hub

https://aws.amazon.com/blogs/security/how-to-build-ci-cd-pipeline-container-vulnerability-scanning-trivy-and-aws-security-hub/

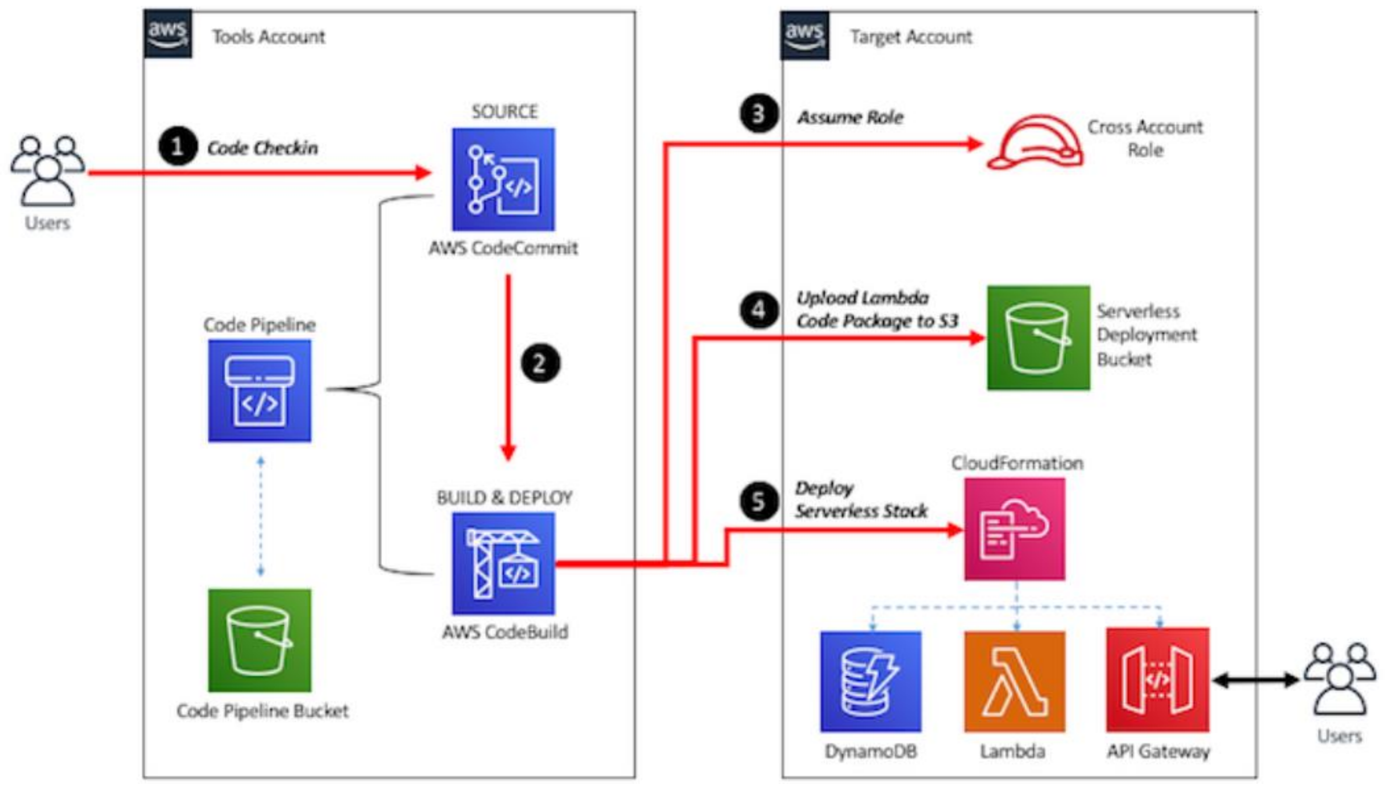
Code Deploy supports linear and Canary deployment for ECS

https://aws.amazon.com/blogs/containers/aws-codedeploy-now-supports-linear-and-canary-deployments-for-amazon-ecs/

ECS supports Blue/Green, Canary, Linnear deployment strategy for application hosted in AWS EC2 instance OR AWS FARGATE with prefefine linear and Canray deployment configuration

Building a CI/CD pipeline for cross-account deployment of an AWS Lambda API with the Serverless Framework

https://aws.amazon.com/blogs/devops/building-a-ci-cd-pipeline-for-cross-account-deployment-of-an-aws-lambda-api-with-the-serverless-framework/



The Tool Account will host the CodePipeline that gets triggered when code is committed to codeCommit repository. CodeBuild leverages the Cross Functional Roles to Create the required resources in Target Account. This architecture is having following advantages over other architecture where codePipeline is inbuild within the target account.

- All pipelines are now located in a centralized account, which consolidates the security controls and grants increased visibility.
- The AWS Identity and Access Management (IAM) permission model is greatly simplified because the pipelines can now share common IAM roles and policies. In addition, there is a clear demarcation between deployment-specific roles that pipelines assume and basic pipeline permissions.
- Logs for all pipelines are located in a single account under Amazon CloudWatch.

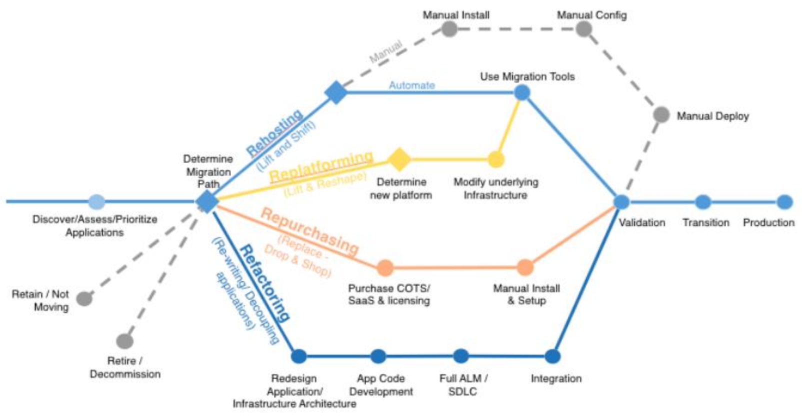
The main component of the codeBuild pipeline is the buildspec.yml which has two parts

1. Install Phase: The install phase defines instructions to install prerequisites, which for this use case is the serverless npm package needed to build and deploy the API using Serverless Framework.
2. Build Phase: This phase builds the resource and deploy the serverless lambda function.

Cloud migration

6 Strategies for Migrating Applications to the Cloud

https://aws.amazon.com/blogs/enterprise-strategy/6-strategies-for-migrating-applications-to-the-cloud/


Manual Install
Manual Config
Manual
Automate
Use Migration Tools
Manual Deploy
Rehosting (Lift and Shift)
Replatforming (Lift & Reshape)
Determine new platform
Modify underlying Infrastructure
Determine Migration Path
Discover/Assess/Prioritize Applications
Repurchasing (Replace - Drop & Shop)
Validation
Transition
Production
Retain / Not Moving
Refactoring (Re-writing/ Decoupling applications)
Purchase COTS/ SaaS & licensing
Manual Install & Setup
Retire / Decommission
Redesign Application/ Infrastructure Architecture
App Code Development
Full ALM / SDLC
Integration

6 R's of Cloud Migration are

1. Rehost – lift and shift
2. Replatforming – make few changes in the architecture for tangible benefits
3. Repurchasing – moving to a different software
4. Refactoring – architecting changing to the architecture to fit cloud land scape.
5. Retire – get rid of the legacy application.
6. Retain – identify and retain certain software that would make more sense for being on-premises.

Migrate Delimited Files from Amazon S3 to an Amazon DynamoDB NoSQL Table Using AWS Database Migration Service and AWS CloudFormation

https://aws.amazon.com/blogs/database/migrate-delimited-files-from-amazon-s3-to-an-amazon-dynamodb-nosql-table-using-aws-database-migration-service-and-aws-cloudformation/

This blog demonstrates how to used DMS to load data from S3 to Dynamo DB. There are four pieces of the couldFormation template

1. DMSReplicationInstance
2. DMSEndpointS3
3. DMSEndpointDynamoDB
4. DMSTaskMigration

Best Practice