Scale usage of AWS KMS keys for AWS Services with multi-region replica and cross-account access

This solution is a set of Terraform modules that provision symmetric customer managed AWS KMS keys for use by the target AWS Services. You can optionally manage the key resource policy for the cross-account access via the AWS Services and the account principals. An additional module is included that supports creating multi-region replica keys in another region. The full set of features is listed here. The solution also provides three example scenarios of how the solution solves common enterprise use cases.

Introduction

AWS Key Management Service (KMS) is a managed service that makes it easy for you to create and control the cryptographic keys that are used to protect your data. AWS KMS is integrated with many AWS Services and integrates with AWS CloudTrail to log use of your KMS keys for auditing, regulatory, and compliance needs. AWS KMS uses an Envelope encryption strategy to protect the keys that encrypt data. Envelope encryption is the practice of encrypting plaintext data with a data key, and then encrypting the data key with a second key, known as the root key. AWS KMS protects the encryption keys by storing and managing them securely. Root keys never leave the AWS KMS unencrypted. Key resource policy along with IAM policies controls the access to the AWS KMS APIs.

AWS KMS keys can be AWS owned, AWS managed or customer managed. Some AWS Services encrypt the data, by default, with an AWS owned key or an AWS managed key. Some AWS Services support customer managed keys. Use a separate customer managed AWS KMS key for each AWS Service that supports it for fine-grained access control. This best practice helps in defining scoped down permissions to the KMS keys and hence the data to the authorized users of the AWS Services.

Many large organizations have an AWS multi-account strategy. A common example is a `Security` account that owns the AWS security resources. An administrator in the `Security` account manages the lifecycle of the AWS security resources. One approach to scale the control of the AWS KMS keys across the accounts is to create the keys in a `Security` account and allowing cross-account access to the trusted accounts in the key resource policy, as shown in the sample resource policy below:

```
{
    "Sid": "Allow use of the key to the cross-account owner",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::<truested-account-id-1>:root",
```

```
            "arn:aws:iam::<trusted-account-id-2>:root",
            "..."
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

A trusted account may further delegate these permissions to IAM principals within their own account using the IAM policies.

```
{
    "Sid": "Allow use of this KMS key from Security Account",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::<trusted-account-id-1>:role/RoleInMyAccount",
            "arn:aws:iam::<trusted-account-id-1>:user/UserInMyAccount"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "arn:aws:kms:<aws-region>:<security-account-id>:key/<key-id>"
}
```

Another approach is to allow access using `kms:ViaService` and `kms:CallerAccount` conditions for the authorized principals of the trusted account(s) in the key resource policy.

```
{
    "Sid": "Allow access through <aws-service> for all principals in the
account(w) that are authorized to use <aws-service>",
    "Effect": "Allow",
    "Principal": {
        "AWS": "*"
    },
    "Action": [
        "kms:ReEncrypt*",
        "kms:Encrypt",
        "kms:DescribeKey",
        "kms:Decrypt",
        "kms:CreateGrant"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:CallerAccount": [
                "<owner-account-id>",
                "<trusted-account-id-1>",
                "<trusted-account-id-2>"
            ],
            "kms:ViaService": "<aws-service>.<aws-region>.amazonaws.com"
        }
    }
}
```

AWS KMS also supports multi-region keys, which are copies of the AWS KMS keys in different AWS Regions. The related multi-region keys have the same key material and key ID, hence cross-region encryption/decryption can occur without re-encryption or cross-region API calls. A multi-region key in each AWS Region has its own alias, tags, key resource policy, grants, and status (enabled/disabled). These are suitable for many common data security scenarios such as disaster recovery, multi-region applications, global data management, etc.

Features

The solution has following features:

- Create the AWS KMS keys along with key resource policy and alias suitable for the target AWS Services.
- Create multi-region replica key with key resource policy and alias in another region.
- Specify one or more IAM roles for the `Administration` of the keys in the owner account.

- Specify zero or more IAM roles for the `Usage` of the keys from the owner account and/or trusted-account(s).
- Specify zero or more trusted accounts for the cross-account root access.
- Enable cross-account access via service authorization pattern.
- Enable multi-region for any of the supported keys.
- Disable key rotation for any of the supported keys.
- Override key resource policy for any of the supported keys.
- Uniform key alias naming and common properties for the keys.
- Uniform names and tags for the provisioned resources.

Prerequisites

- The target AWS Account(s) and AWS Region(s) are identified.
- The AWS User/Role executing the Terraform scripts must have permissions to provision the target resources in the owner account.
- The Terraform CLI (`version = ">= 1.1.9"`) is installed.
- Terraform backend provider and state locking providers are identified and bootstrapped.
    - An example bootstrap module/example is provided that provisions an Amazon S3 bucket for Terraform state storage and Amazon DynamoDB table for Terraform state locking.
        - The Amazon S3 bucket name must be globally unique.
- A unique project code name e.g., `appx` is identified that will be used to uniformly name the key aliases.
- Uniform resource tagging scheme is identified.
    - The examples use only two tags: `Env` and `Project`

Usage

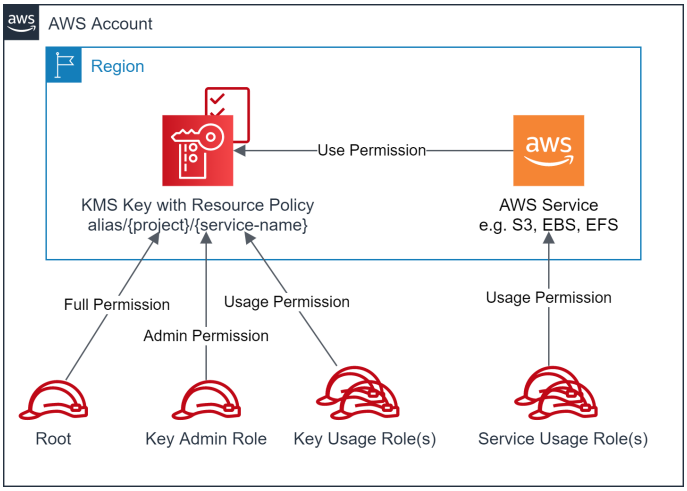
- Use the module via GitHub source or copy the module into your repository.
- Incorporate the module in your administration CI/CD pipeline as appropriate.
- For multi-account strategy, this module should be used by the `Security` account administrator or the pipeline.

Scenarios

This solution primarily supports the following scenarios though many other scenarios are possible.

Scenario 1: Create single-region AWS KMS key(s) in an account for the target AWS Services

Create one or more single-region AWS KMS keys in the owner account along with key resource policies and aliases that can be used by the target AWS Services.

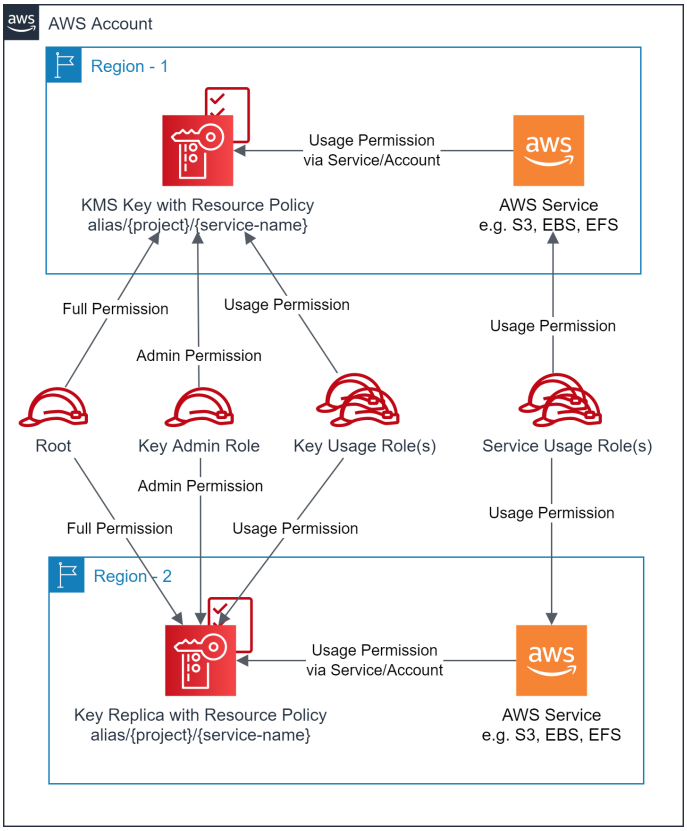


- Account owner has full access to the key(s)
- Key Admin role has administrative access to the key(s)
- Key Usage role(s) have the usage access to the key(s)
- Target AWS Service usage role(s) have the usage access to the key via the target AWS Service.

Refer examples/kms/scenario1 to execute this scenario.

Scenario 2: Create multi-region AWS KMS key(s) in the primary region and multi-region replica key in another region(s)

Create one or more multi-region AWS KMS keys along with key resource policies and aliases in the primary region, along with multi-region replica key(s) in another region(s). The target AWS Service in the secondary region(s) will be able to use the Key replica via the known alias.

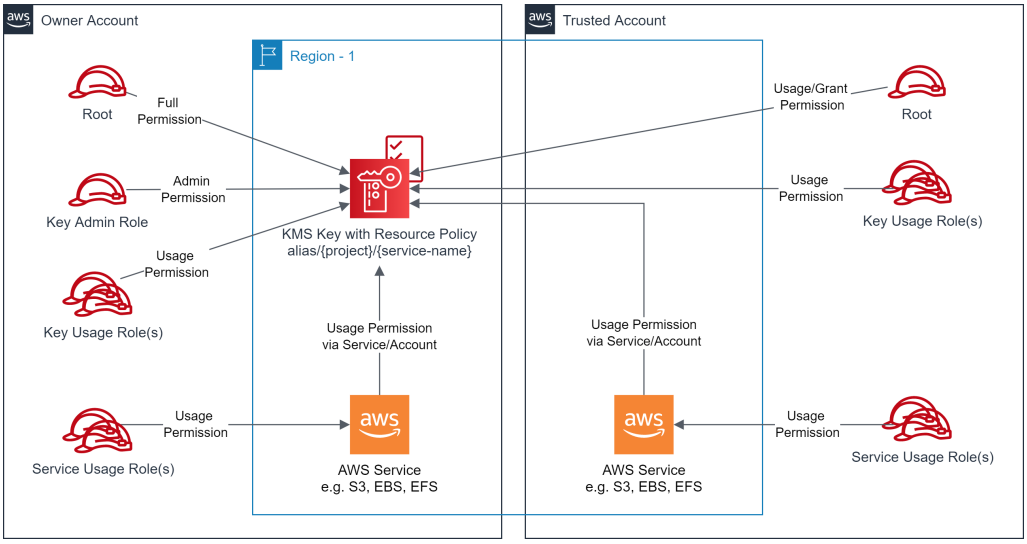


- Account owner has full access to the key(s) and replica key(s)
- Key Admin role has administrative access to the key(s) and replica key(s)
- Key Usage role(s) have the usage access to the key(s) and replica key(s)
- Target AWS Service usage role(s) have the usage access to the key or replica key via the target AWS Service in the respective region.

Refer examples/kms/scenario2 to execute this scenario.

Scenario 3: Create AWS KMS key(s) in the owner account and allow cross-account access via AWS Services and the account principals

Create one or more AWS KMS keys in the one account. Allow principal(s) and AWS Services in the trusted account(s) to use the AWS KMS keys in their account.



- Account owner has full access to the key(s)
- Key Admin role in the owner account has administrative access to the key(s)
- Key Usage role(s) in the owner account have the usage access to the key(s)
- Target AWS Service usage role(s) in the owner account have the usage access to the key via the target AWS Service in the owner account
- Trusted account `root` has the usage and grand access to the Key(s). It can delegate usage access to other principals in the trusted account.
- Key Usage role(s) in the trusted account have the usage access to the key(s)
- Target AWS Service usage role(s) in the trusted account have the usage access to the key via the target AWS Service in the trusted account

Refer examples/kms/scenario3 to execute this scenario.

Supported Services

This set of modules supports creating the AMS KMS key along with key resource policy and alias that can be used by the following AWS Services.

- Amazon S3
- Amazon EBS
- Amazon EFS
- Amazon RDS
- Amazon DynamoDB
- AWS Lambda
- Amazon CloudWatch Logs
- Amazon SNS
- Amazon SQS

- AWS Backup
- AWS Secrets Manager
- AWS Systems Manager Parameter Store
- AWS Systems Manager Session Manager
- Amazon Kinesis
- AWS Glue
- AWS ACM
- Amazon MWAA
- Amazon ECR
- Amazon EKS

Future Enhancements

- The solution supports Symmetric KMS keys. It can be enhanced to support Asymmetric KMS keys.
- The solution can be enhanced to support more AWS Services.
- The solution support common use cases and sufficient key resource policies. It can be enhanced to further scope-down the policies.

Security

See CONTRIBUTING for more information.

License