Cremit

Credential 살펴보기, Focus on AWS

발표자 소개

김동현, Ben Kim

Cremit, Founder - CEO
Sendbird, Watcha, …
Hacking Conference
Hacking Competition
10 yrs+ Security Career

Cremit Jobs

- Product, DevSecOps
- HR, Investor Relationship
- Sales, Marketing

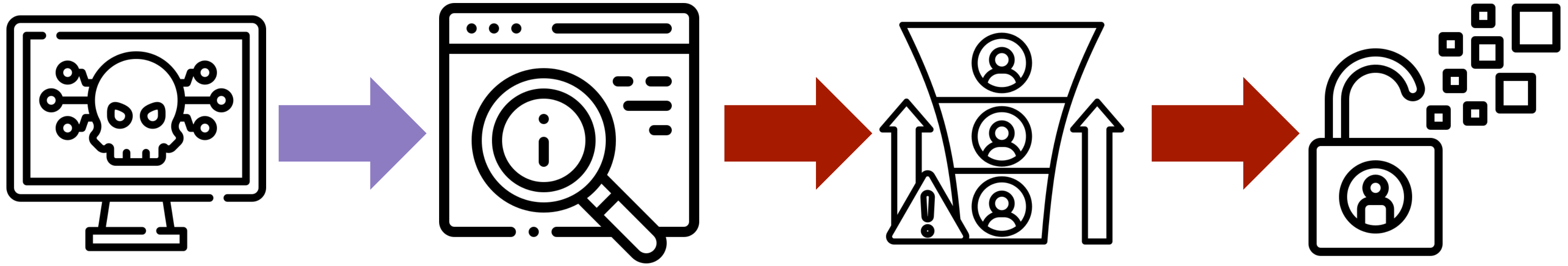
ben@cremit.io, hi@cremit.io

Cremit

Credential

What is Credential

ID/Password API Key Personal SSL/TLS Data Encryption



악성코드 / 취약점
외부 획득

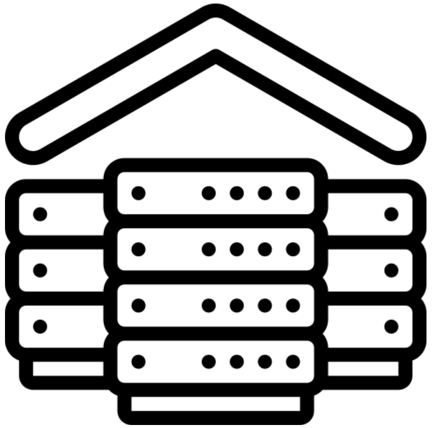
크리덴셜 획득

권한 상승

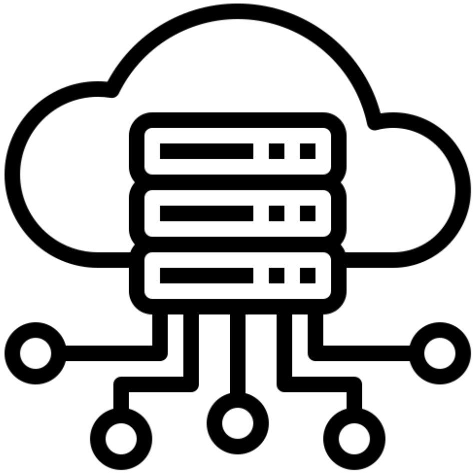
해킹사고

The Shift

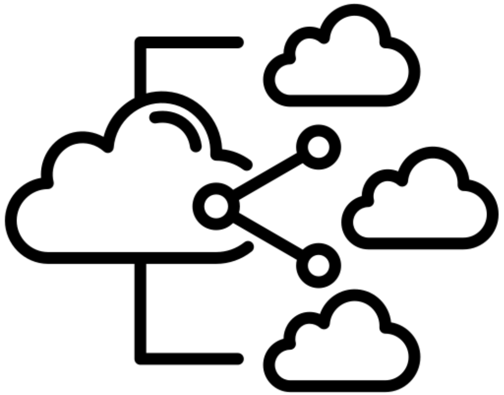
The transition to cloud & multi-cloud



On-Premise



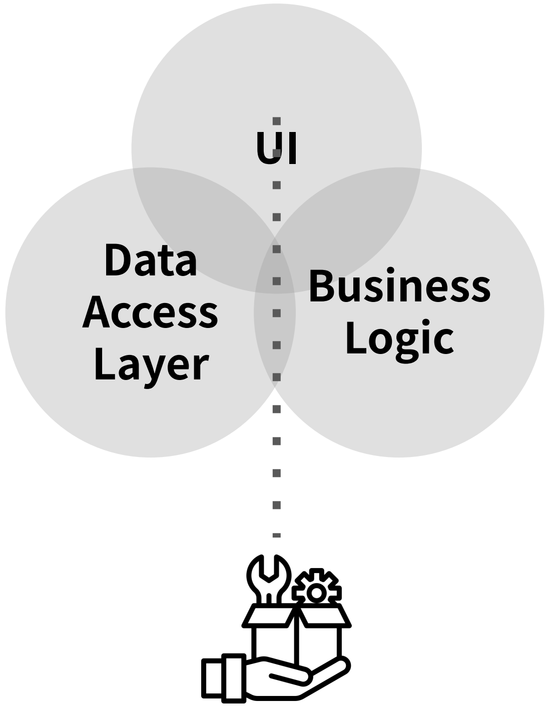
Cloud

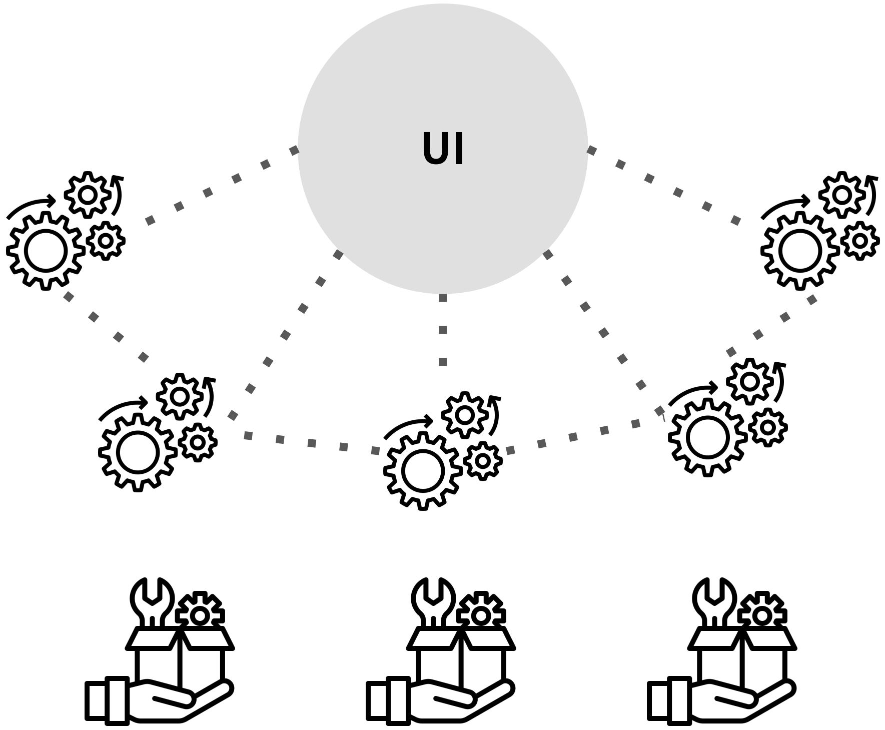


Work on Cloud

The Shift

The transition service architecture to Micro Service





The Problem

12M

New Credential detected(+28%)
공개된 Github 코드에서 탐색된 횟수

Microsoft 2023, Sep. 22

마이크로소프트의 개발자가 Github 코드저장소에 업로드 한 자료 내 포함되어 있는 **Azure SAS Token(Credential)**이 노출되어 38TB가 노출된 사고로, 30,000개 이상의 내부 대화 내용, OpenAI의 영업 비밀, 다수의 내부 인프라 접근 정보가 포함되었습니다.
SAS Token은 일반적인 웹사이트 주소와 같아, 해당 내용이 민감한 Credential인지 구분하기에 매우 어렵습니다.

Uber 2022, Sep. 15

해커는, 하드 코딩된 **관리자의 자격증명(Credential)**을 Thycotic 로그에서 발견하여 우버 내부의 관리 도구에 접근하여, 내부 전체 계정을 탈취하였습니다.

okta 2023, Oct. 29

해커는, 옥타의 고객 지원 시스템에 존재하는 HAR 파일을 획득, 해당 파일을 통해 **직원의 Credential을 탈취하여 옥타 내부의 고객 정보와 각종 Access Key**를 탈취하였습니다.

Cloudflare 2023, Nov. 14

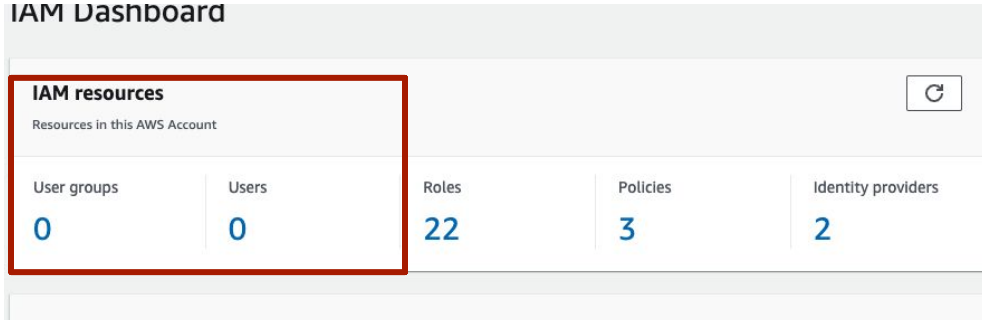
위의 옥타 해킹사건에서 **유출된 Credential을 통해, Cloudflare의 내부 서버에 국가기관 해커가 침투**하였습니다. Cloudflare는 Credential 유출에 대한 조치를 하지 않았고, **5000개의 Credential 교체 작업**, 포렌식, 회사 시스템 전체 재부팅을 진행하였습니다.

Cremit

AWS IAM Credential

Management Credential on AWS

IAM?

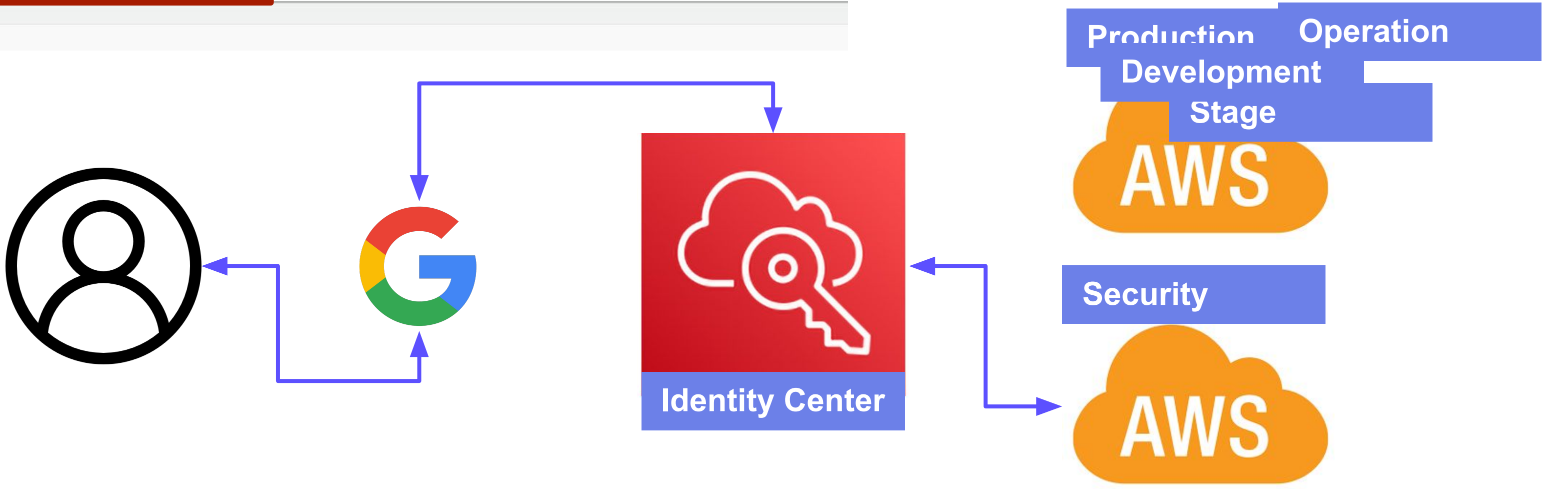


IAM Dashboard

IAM resources

Resources in this AWS Account

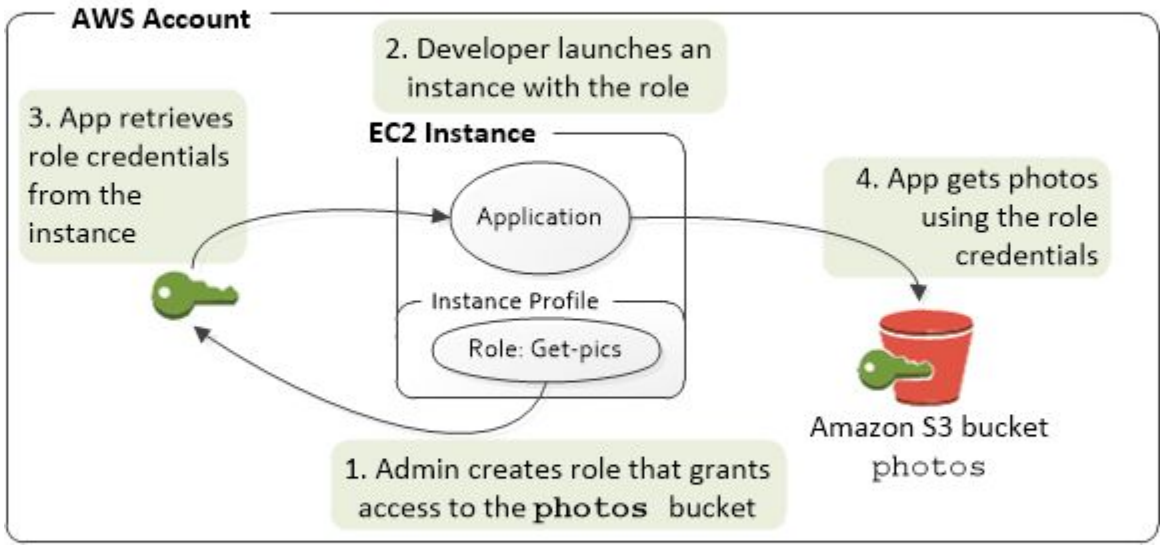
User groups	Users	Roles	Policies	Identity providers
0	0	22	3	2



Cremit

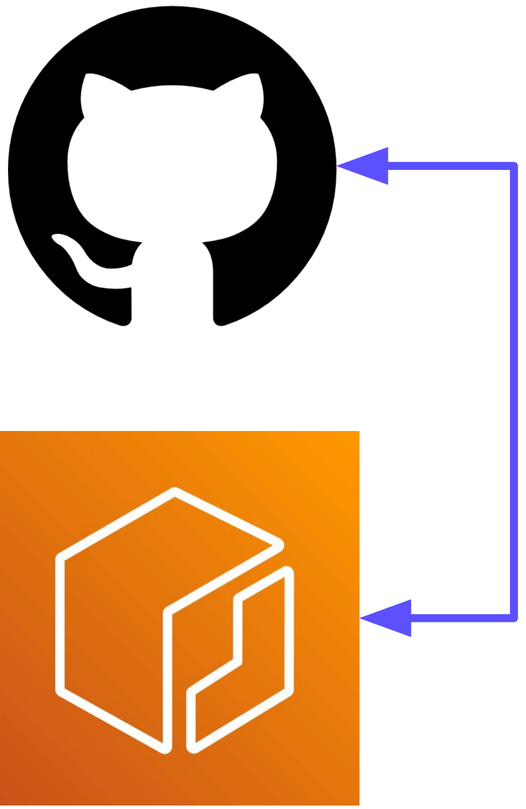
Management Credential on AWS

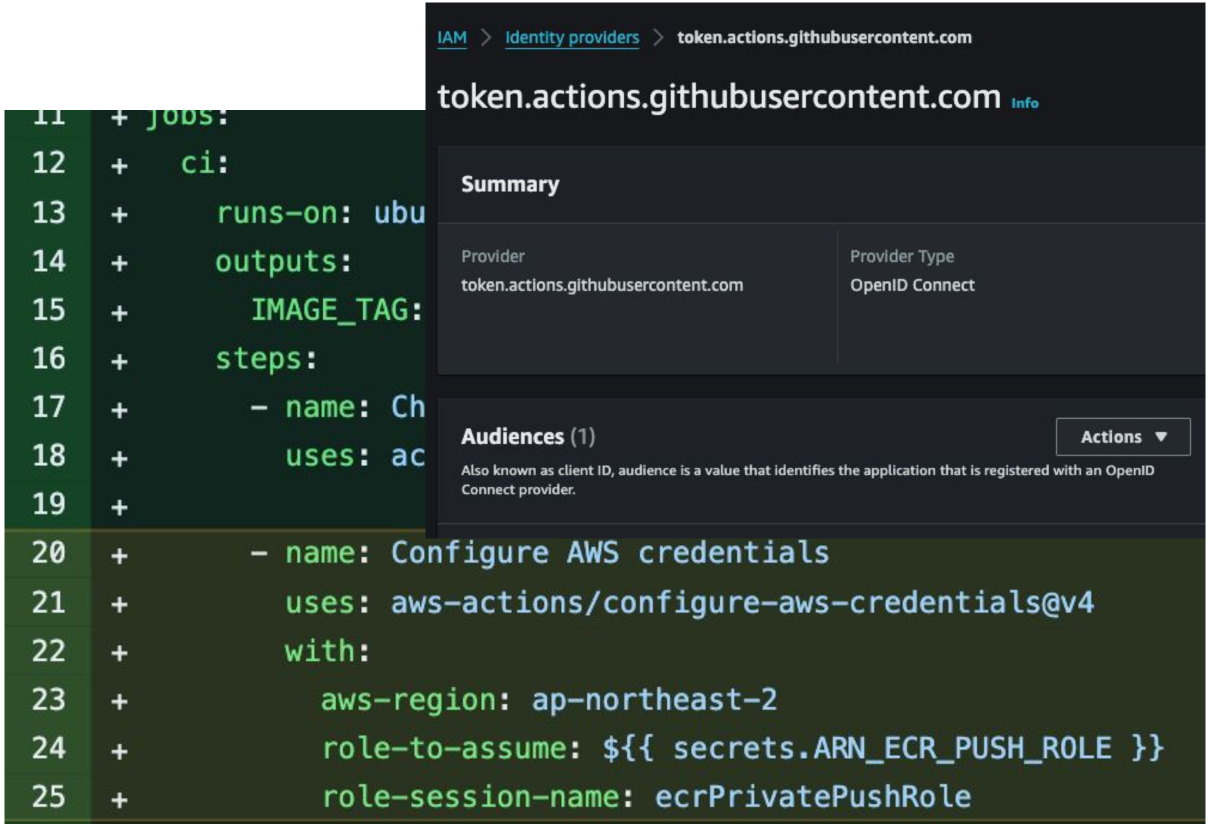
EC2 <> AWS Resources


AWS Account
2. Developer launches an instance with the role
3. App retrieves role credentials from the instance
EC2 Instance
Application
4. App gets photos using the role credentials
Instance Profile
Role: Get-pics
Amazon S3 bucket photos
1. Admin creates role that grants access to the photos bucket

IAM Credential

3rd Party <> AWS Resources





IAM > Identity providers > token.actions.githubusercontent.com

token.actions.githubusercontent.com Info

Summary

Provider	Provider Type
token.actions.githubusercontent.com	OpenID Connect

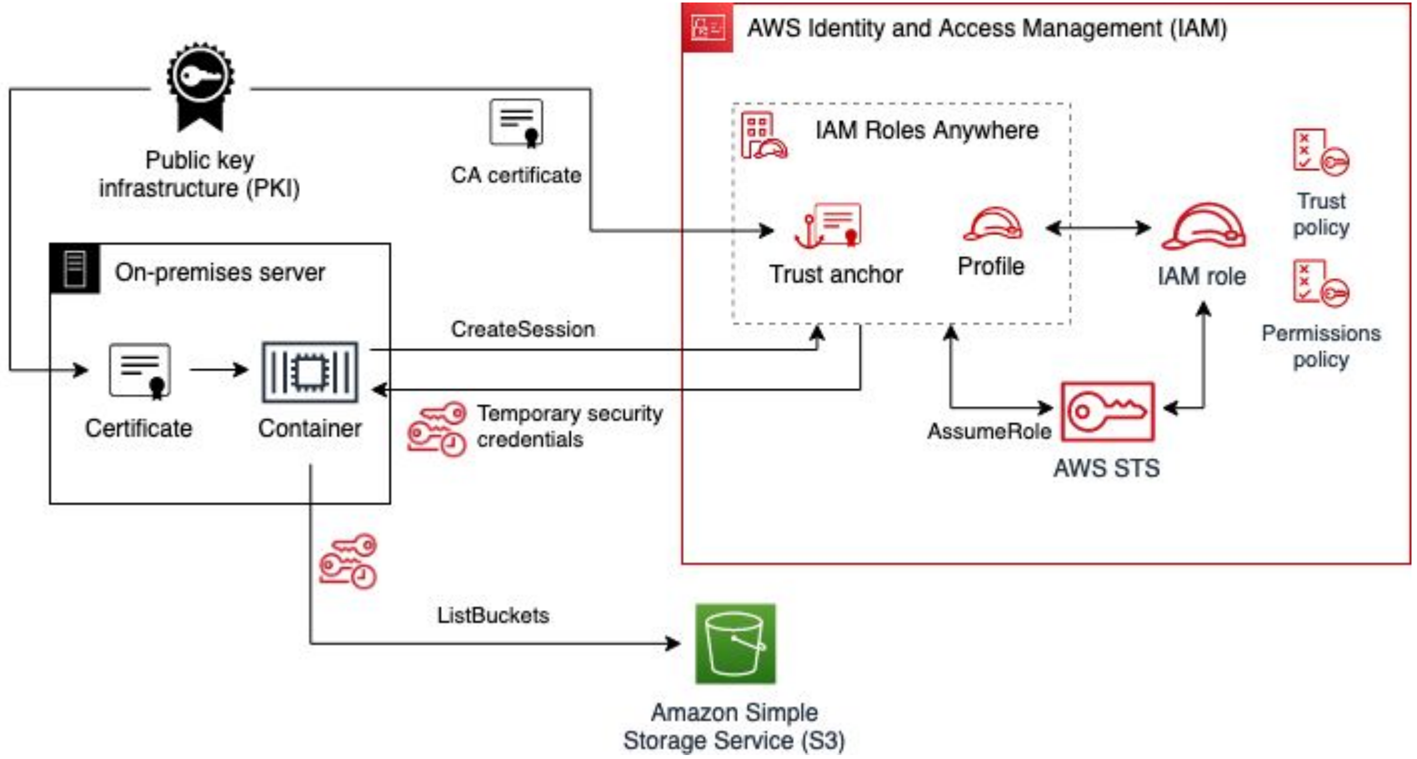
Audiences (1)

Also known as client ID, audience is a value that identifies the application that is registered with an OpenID Connect provider.

```
+ jobs:
+   ci:
+     runs-on: ubu
+     outputs:
+       IMAGE_TAG:
+     steps:
+       - name: Ch
+         uses: ac
+
+       - name: Configure AWS credentials
+         uses: aws-actions/configure-aws-credentials@v4
+         with:
+           aws-region: ap-northeast-2
+           role-to-assume: ${{ secrets.ARN_ECR_PUSH_ROLE }}
+           role-session-name: ecrPrivatePushRole
```

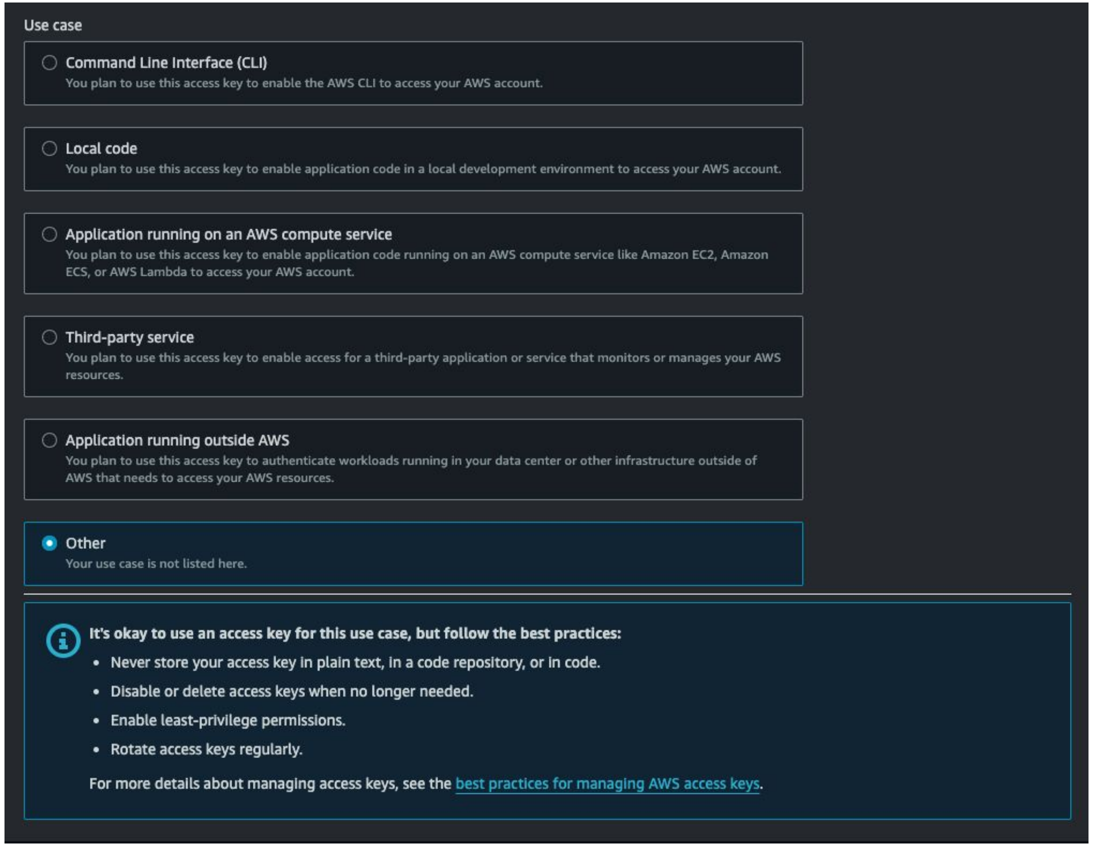
IAM Credential

Outside Cloud (e.g On-premise)



Management Credential on AWS

Do not create Access Key



Use case

○ Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

○ Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.

○ Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

○ Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

○ Application running outside AWS
You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.

[✓ Other]
Your use case is not listed here.

It's okay to use an access key for this use case, but follow the best practices:

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access keys when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the best practices for managing AWS access keys.

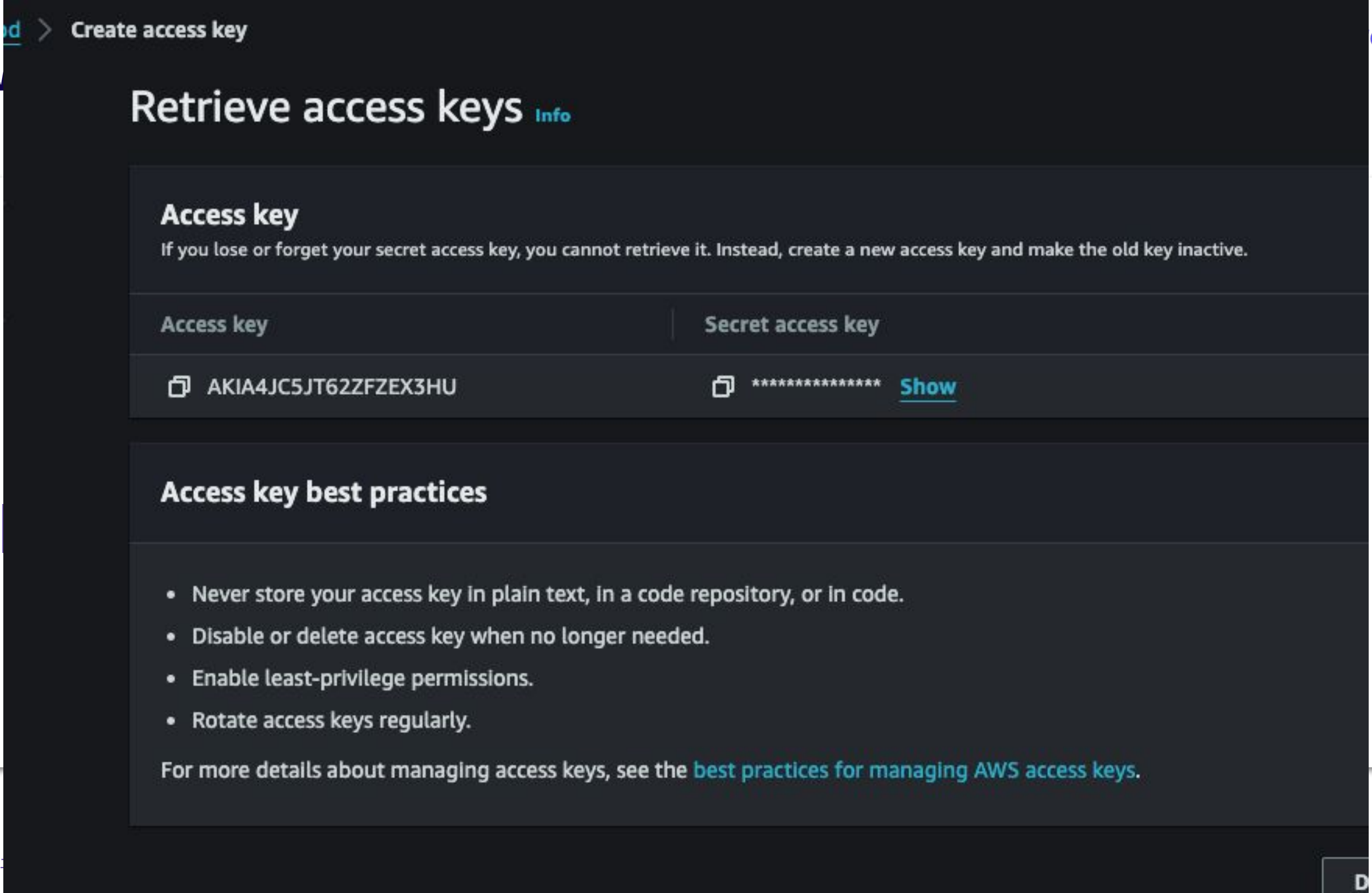
03

Cremit

AWS Access Key & Secret Key

AW

ASIA



od > Create access key

Retrieve access keys Info

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIA4JC5JT62ZFZEX3HU	*************** Show

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the best practices for managing AWS access keys.

emit

AWS Credential Pattern

ASIA vs AKIA?

AKIAOSF….

Wjalr…

Long-Term

ASIA….

9drT….

AqoXd////…

Temporary

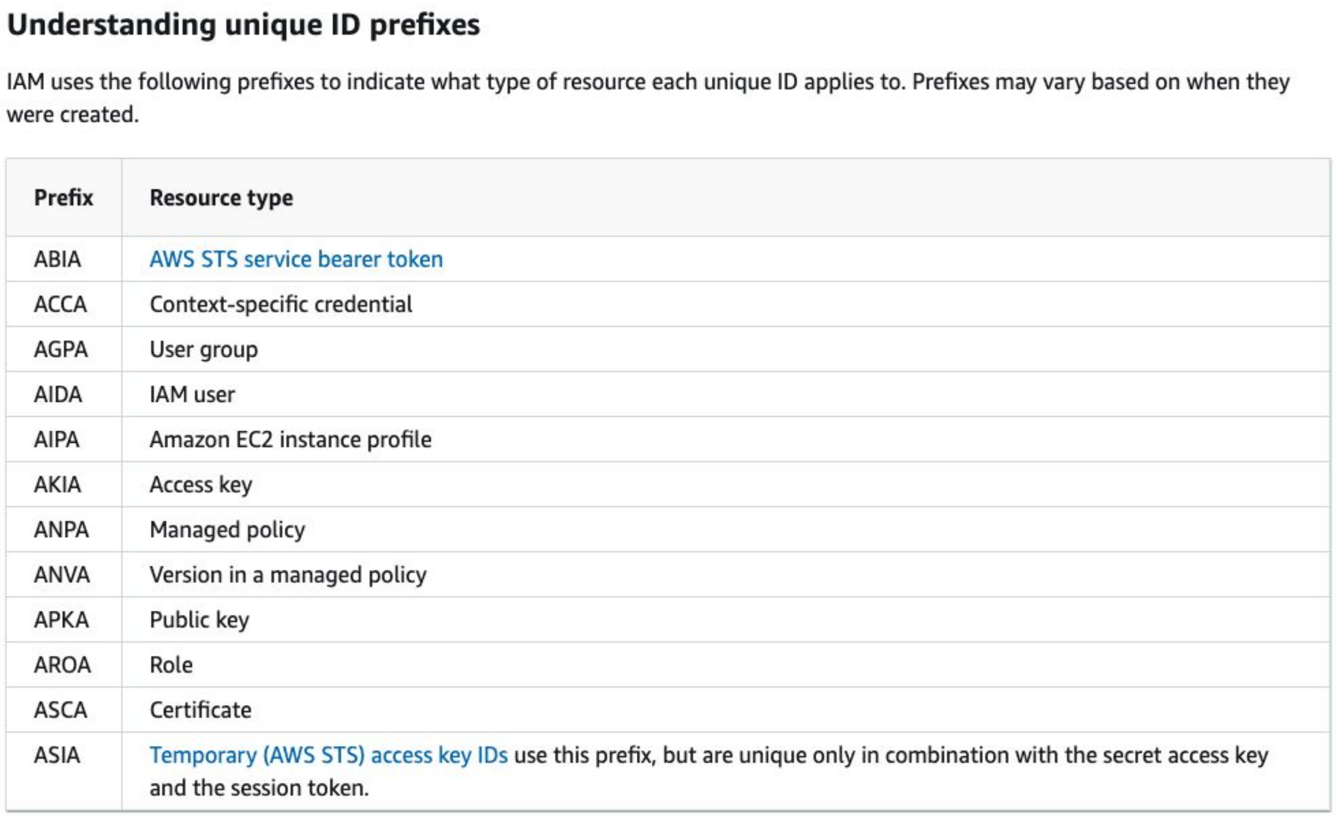
What Else?

ABIA….

Cremit

AWS Credential Pattern

ASIA vs AKIA? https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-unique-ids



Understanding unique ID prefixes

IAM uses the following prefixes to indicate what type of resource each unique ID applies to. Prefixes may vary based on when they were created.

Prefix	Resource type
ABIA	AWS STS service bearer token
ACCA	Context-specific credential
AGPA	User group
AIDA	IAM user
AIPA	Amazon EC2 instance profile
AKIA	Access key
ANPA	Managed policy
ANVA	Version in a managed policy
APKA	Public key
AROA	Role
ASCA	Certificate
ASIA	Temporary (AWS STS) access key IDs use this prefix, but are unique only in combination with the secret access key and the session token.

AWS Credential Pattern

AORA Trick

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": {
        "AWS": "arn:aws:iam::607481581596:role/service-role/abctestrole"
      },
      "Effect": "Deny",
      "Action": ["s3:GetObject"],
      "Resource": "arn:aws:s3:::YOUR_BUCKET_NAME_HERE/*"
    }
  ]
}
```

AWS Credential Pattern

AROA Trick

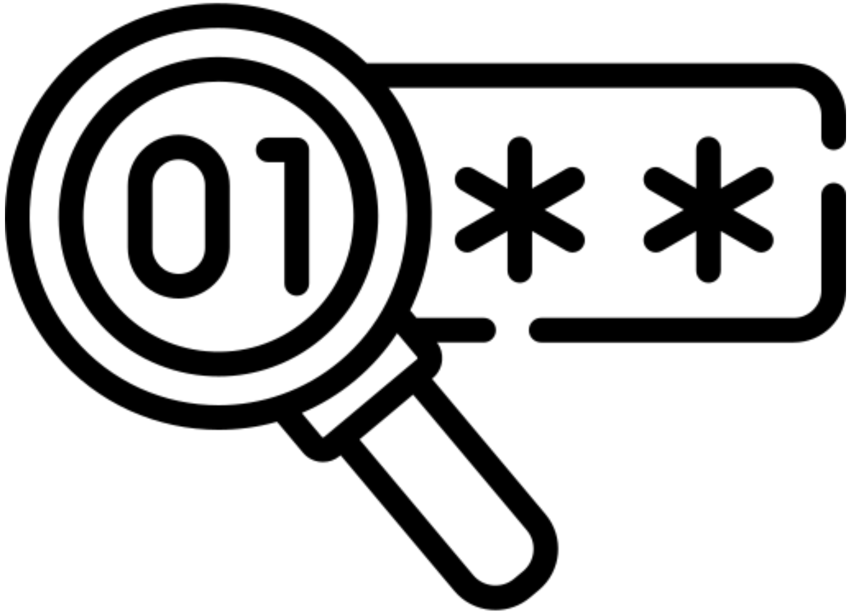
aws iam get-role --role-name "<you role here>"

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": {
        "AWS": "AROAJMD24IEMKTX6BABJI"
      },
      "Effect": "Deny",
      "Action": ["s3:GetObject"],
      "Resource": "arn:aws:s3:::YOUR_BUCKET_NAME_HERE/*"
    }
  ]
}
```

Cremit

AWS Credential Pattern

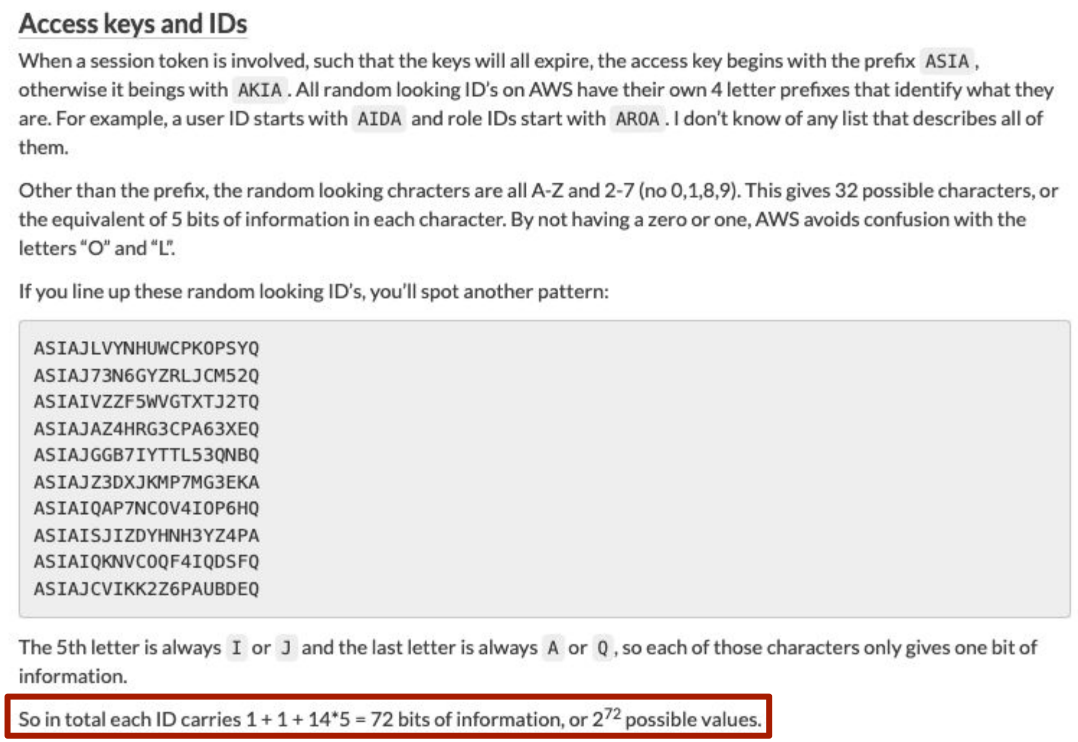
AORA Trick



AWS Credential Pattern

Access Key ID Deep Dive

https://summitroute.com/blog/2018/06/20/aws_security_credential_formats/



Access keys and IDs

When a session token is involved, such that the keys will all expire, the access key begins with the prefix `ASIA`, otherwise it beings with `AKIA`. All random looking ID's on AWS have their own 4 letter prefixes that identify what they are. For example, a user ID starts with `AIDA` and role IDs start with `AROA`. I don't know of any list that describes all of them.

Other than the prefix, the random looking chracters are all A-Z and 2-7 (no 0,1,8,9). This gives 32 possible characters, or the equivalent of 5 bits of information in each character. By not having a zero or one, AWS avoids confusion with the letters "O" and "L".

If you line up these random looking ID's, you'll spot another pattern:

```
ASIAJLVYNHUWCPKOPSYQ
ASIAJ73N6GYZRLJCM52Q
ASIAIVZZF5WVGTXTJ2TQ
ASIAJAZ4HRG3CPA63XEQ
ASIAJGGB7IYTTL53QNBQ
ASIAJZ3DXJKMP7MG3EKA
ASIAIQAP7NCOV4IOP6HQ
ASIAISJIZDYHNH3YZ4PA
ASIAIQKNVCOQF4IQDSFQ
ASIAJCVIKK2Z6PAUBDEQ
```

The 5th letter is always `I` or `J` and the last letter is always `A` or `Q`, so each of those characters only gives one bit of information.

So in total each ID carries 1 + 1 + 14*5 = 72 bits of information, or 2^{72} possible values.

AWS Credential Pattern

Access Key ID Deep Dive

https://awsteele.com/blog/2020/09/26/aws-access-key-format.html

```
$ aws sts get-access-key-info --access-key-id ASIAY34FZKBOKMUTVV7A --query Account
"609629065308"

$ aws sts get-access-key-info --access-key-id ASIAY34FZKBNKMUTVV7A --query Account
"609629065306"
```

AWS Credential Pattern

Access Key ID Deep Dive

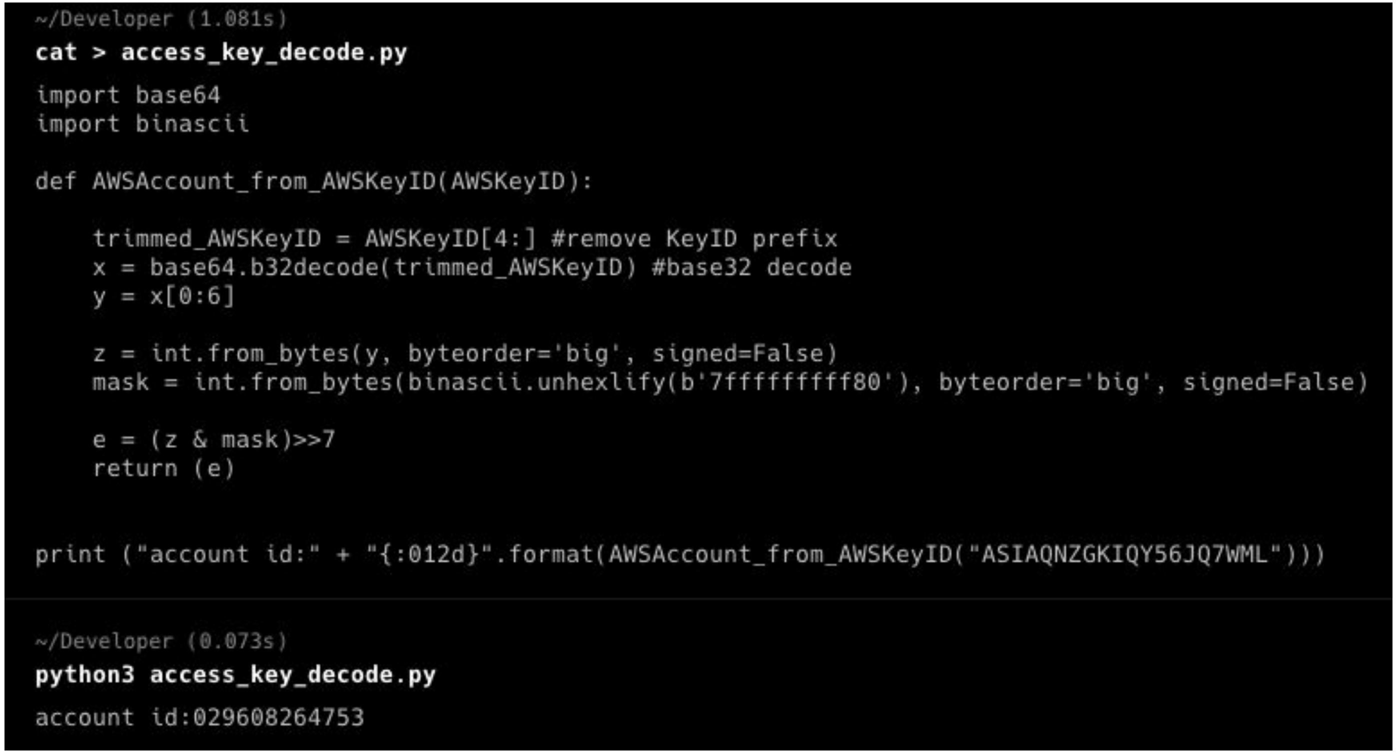
https://medium.com/@TalBeerySec/a-short-note-on-aws-key-id-f88cc4317489

```
import base64
import binascii

def AWSAccount_from_AWSKeyID ( AWSKeyID ):

  Trimmed_AWSKeyID = AWSKeyID[ 4 :] #remove KeyID prefix
   x = base64.b32decode(trimmed_AWSKeyID) #base32 decode
   y = x[ 0 : 6 ]

  z = int .from_bytes(y, byteorder= 'big' , signed= False )
  mask = int .from_bytes(binascii.unhexlify( b'7fffffffff80' ), byteorder= 'big' , signed= False )

  e = (z & mask)>> 7
  return (e)


print ( "Account ID:" + "{:012d}" . 형식 (AWSAccount_from_AWSKeyID( "ASIAQNZGKIQY56JQ7WML" )))
```

AWS Credential Pattern

Access Key ID Deep Dive

https://medium.com/@TalBeerySec/a-short-note-on-aws-key-id-f88cc4317489

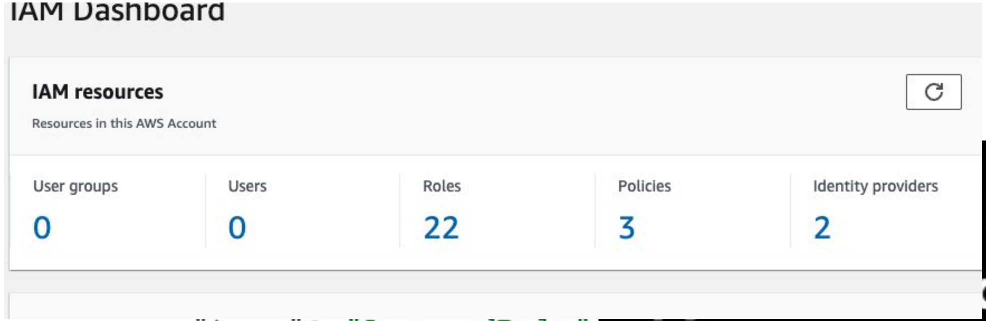


```
~/Developer (1.081s)
cat > access_key_decode.py
import base64
import binascii

def AWSAccount_from_AWSKeyID(AWSKeyID):

    trimmed_AWSKeyID = AWSKeyID[4:] #remove KeyID prefix
    x = base64.b32decode(trimmed_AWSKeyID) #base32 decode
    y = x[0:6]

    z = int.from_bytes(y, byteorder='big', signed=False)
    mask = int.from_bytes(binascii.unhexlify(b'7fffffffff80'), byteorder='big', signed=False)

    e = (z & mask)>>7
    return (e)


print ("account id:" + "{:012d}".format(AWSAccount_from_AWSKeyID("ASIAQNZGKIQY56JQ7WML")))
```

```
~/Developer (0.073s)
python3 access_key_decode.py
account id:029608264753
```

AWS Credential Pattern

Access Key ID Deep Dive

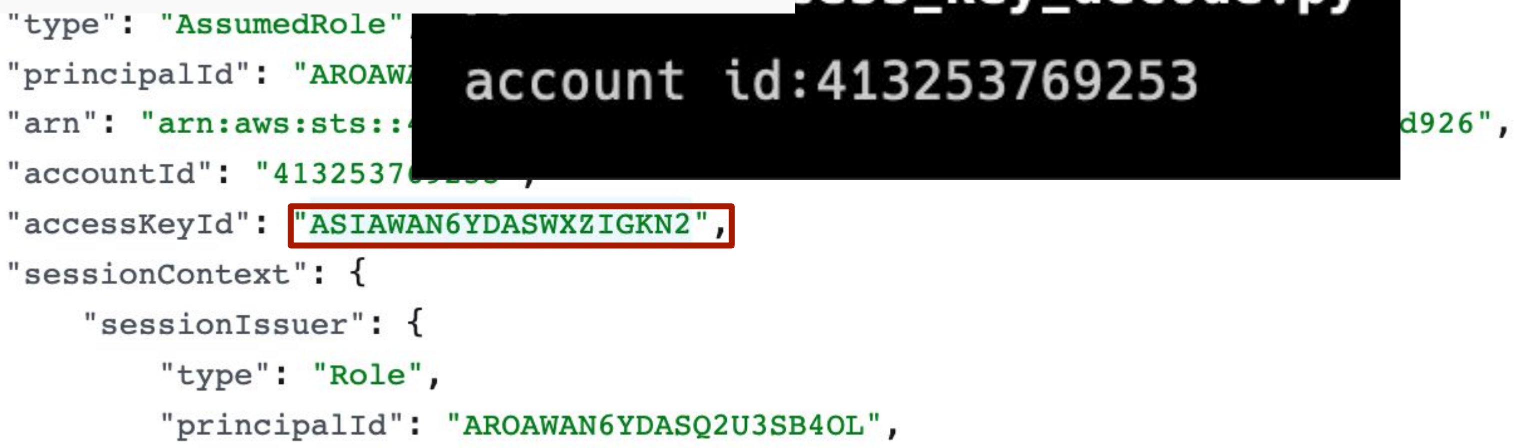


IAM Dashboard

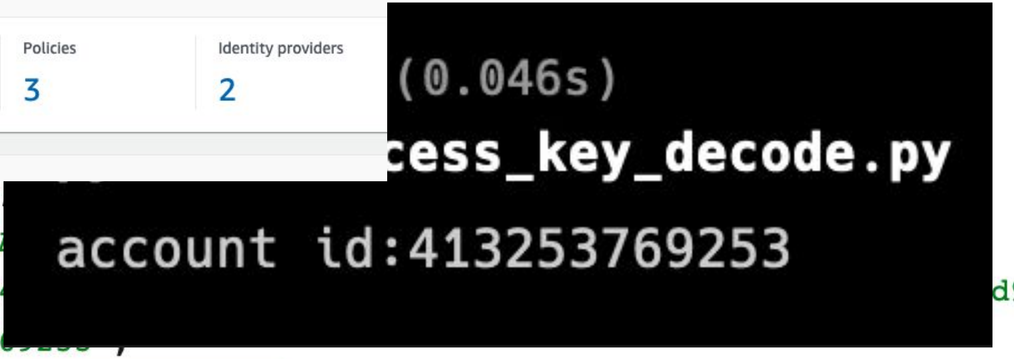
IAM resources

Resources in this AWS Account

User groups	Users	Roles	Policies	Identity providers
0	0	22	3	2



```
"type": "AssumedRole",
"principalId": "AROAW
"arn": "arn:aws:sts::4                                        d926",
"accountId": "4132537
"accessKeyId": "ASIAWAN6YDASWXZIGKN2",
"sessionContext": {
    "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAWAN6YDASQ2U3SB4OL",
```



```
(0.046s)
cess_key_decode.py
account id:413253769253
```

04

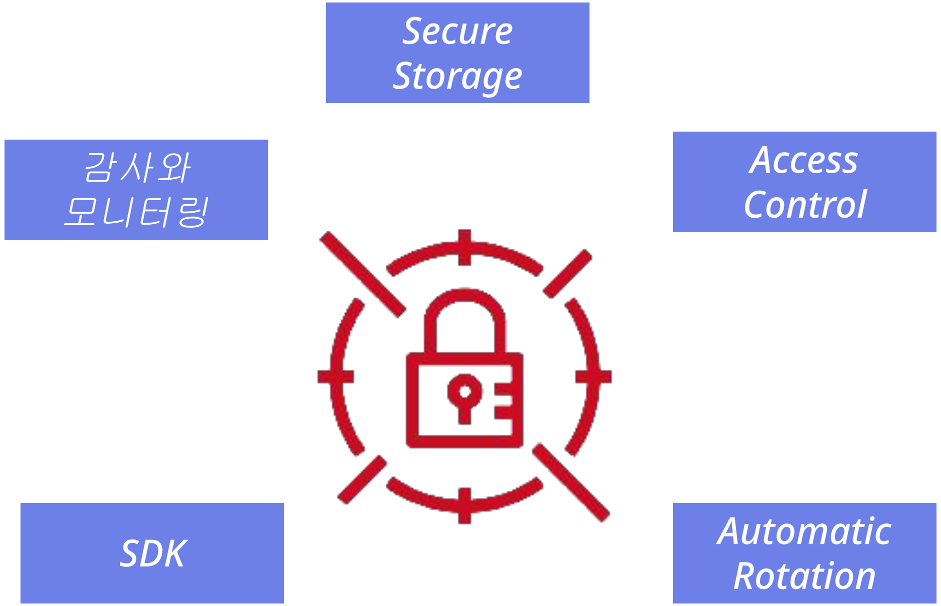
Cremit

Secret Manager Use Cases

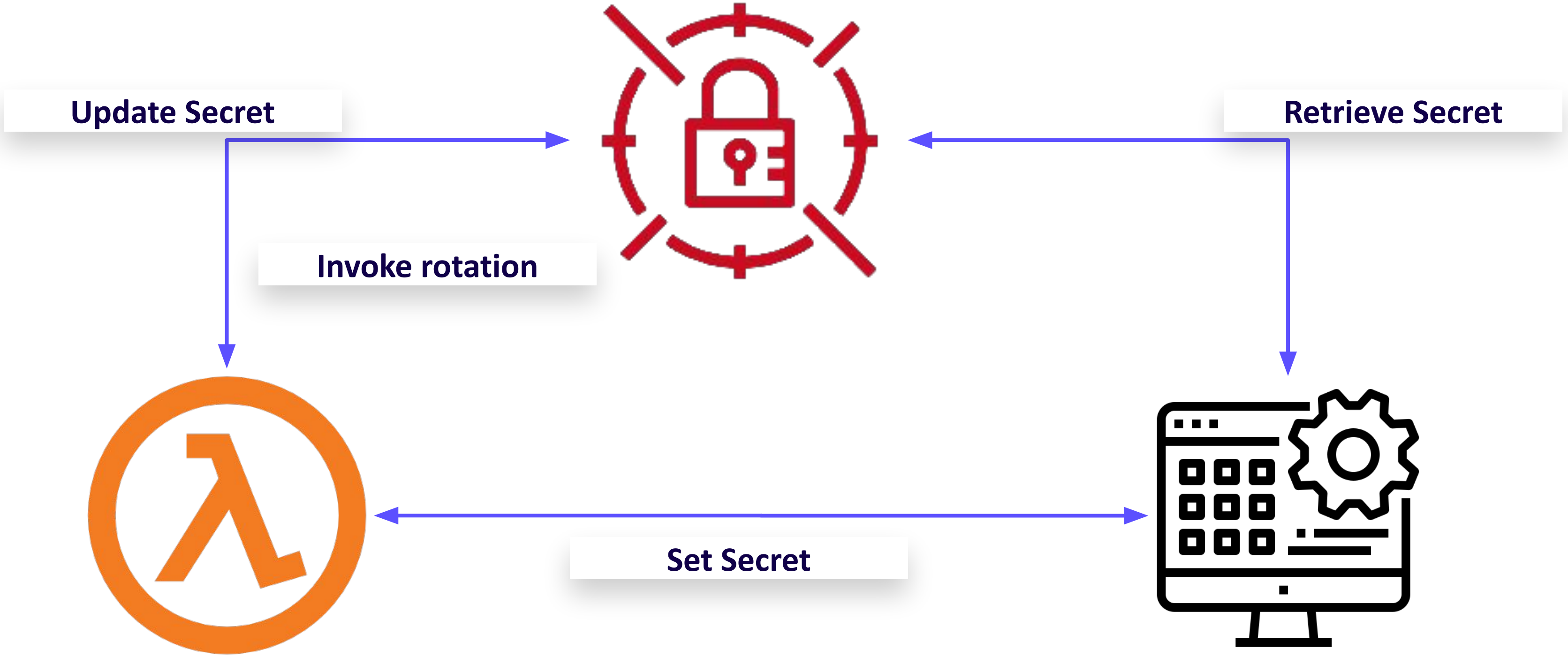
Cremit

Secret Manager use case

Secret Manager Benefits



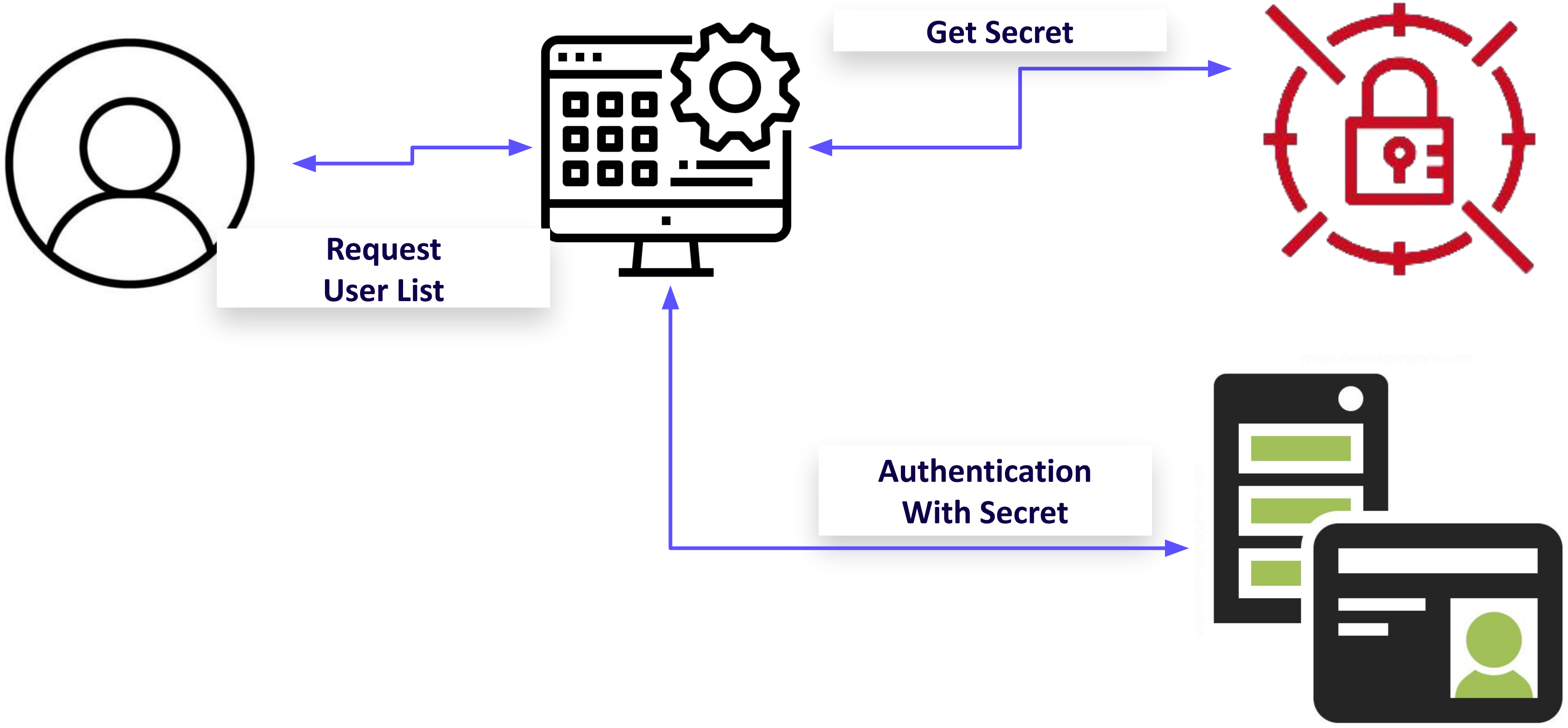
How Secrets Manager Rotation Works



Cremit

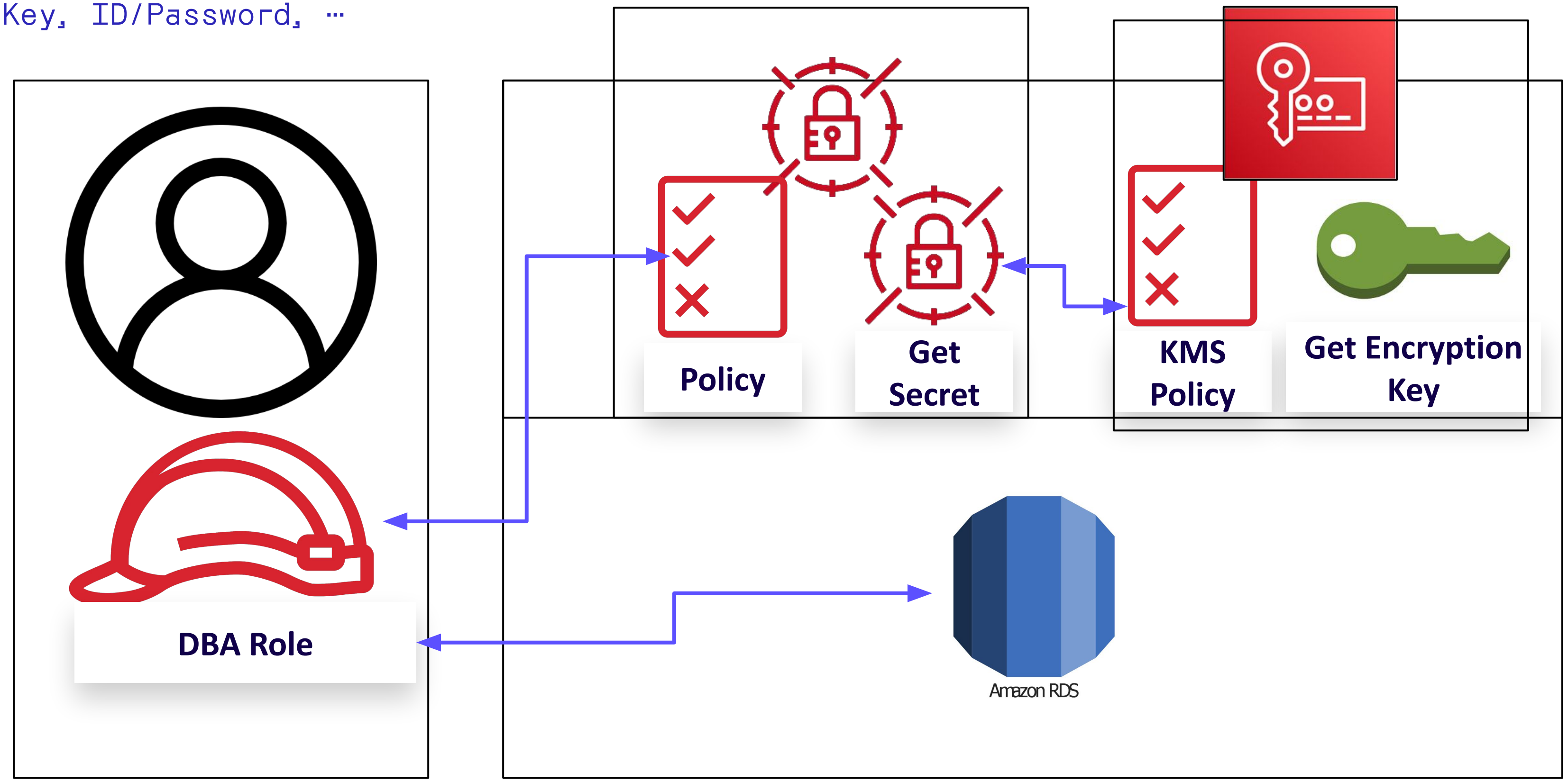
Generic Secret Manager Use Case

API Key, ID/Password, …



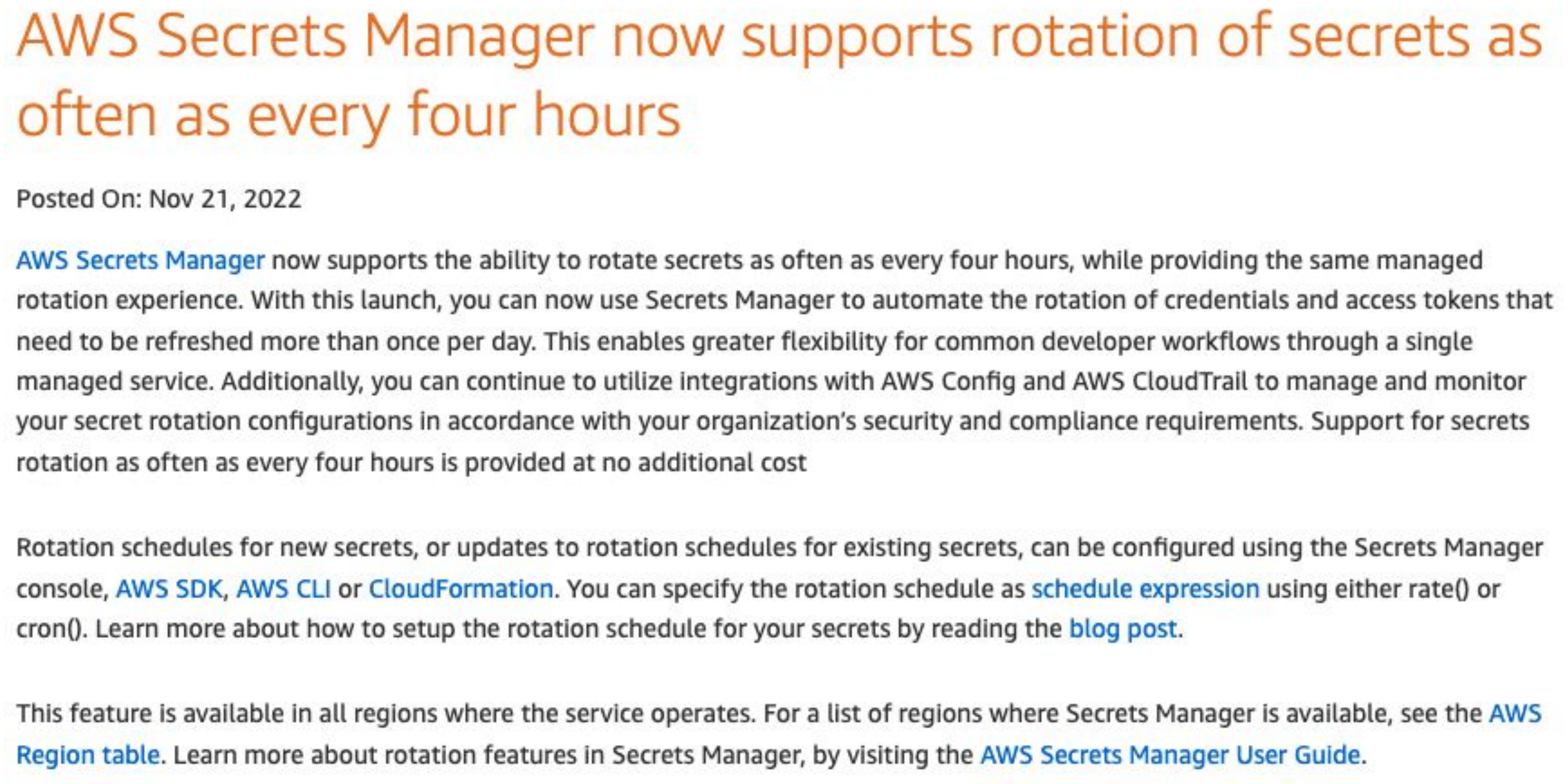
Secret Manager / Role / Get DB Secrets

API Key, ID/Password, …



Rotation Every four hours

Secret manager news



AWS Secrets Manager now supports rotation of secrets as often as every four hours

Posted On: Nov 21, 2022

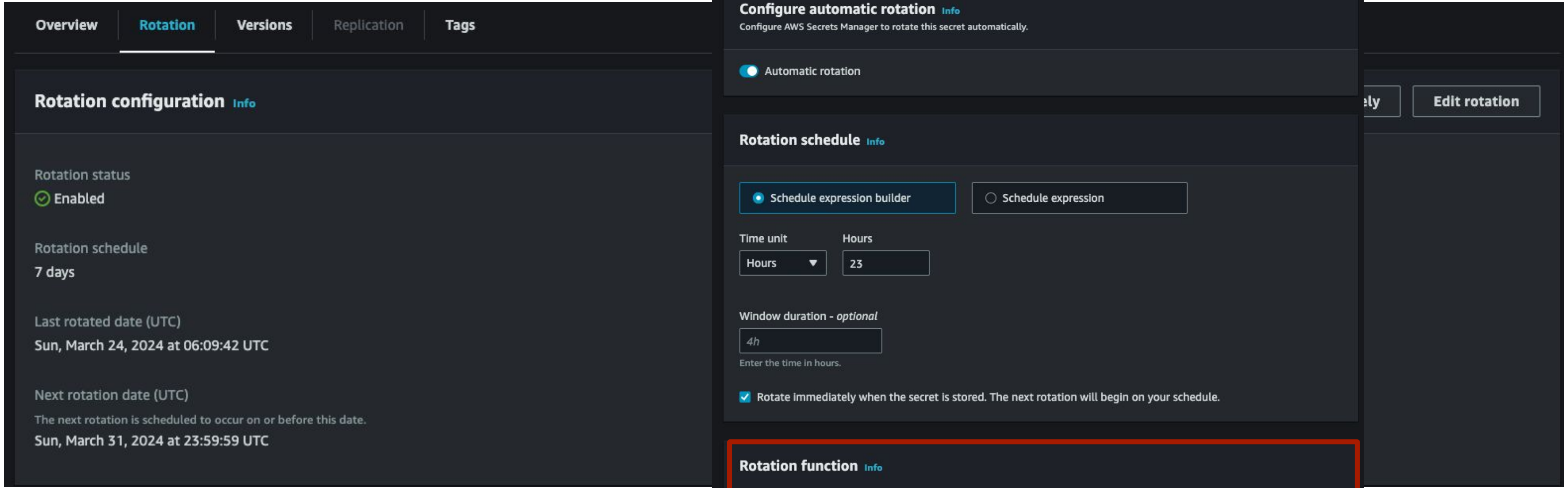
AWS Secrets Manager now supports the ability to rotate secrets as often as every four hours, while providing the same managed rotation experience. With this launch, you can now use Secrets Manager to automate the rotation of credentials and access tokens that need to be refreshed more than once per day. This enables greater flexibility for common developer workflows through a single managed service. Additionally, you can continue to utilize integrations with AWS Config and AWS CloudTrail to manage and monitor your secret rotation configurations in accordance with your organization's security and compliance requirements. Support for secrets rotation as often as every four hours is provided at no additional cost

Rotation schedules for new secrets, or updates to rotation schedules for existing secrets, can be configured using the Secrets Manager console, AWS SDK, AWS CLI or CloudFormation. You can specify the rotation schedule as schedule expression using either rate() or cron(). Learn more about how to setup the rotation schedule for your secrets by reading the blog post.

This feature is available in all regions where the service operates. For a list of regions where Secrets Manager is available, see the AWS Region table. Learn more about rotation features in Secrets Manager, by visiting the AWS Secrets Manager User Guide.

Rotation Every four hours

Secret Manager automatic rotation



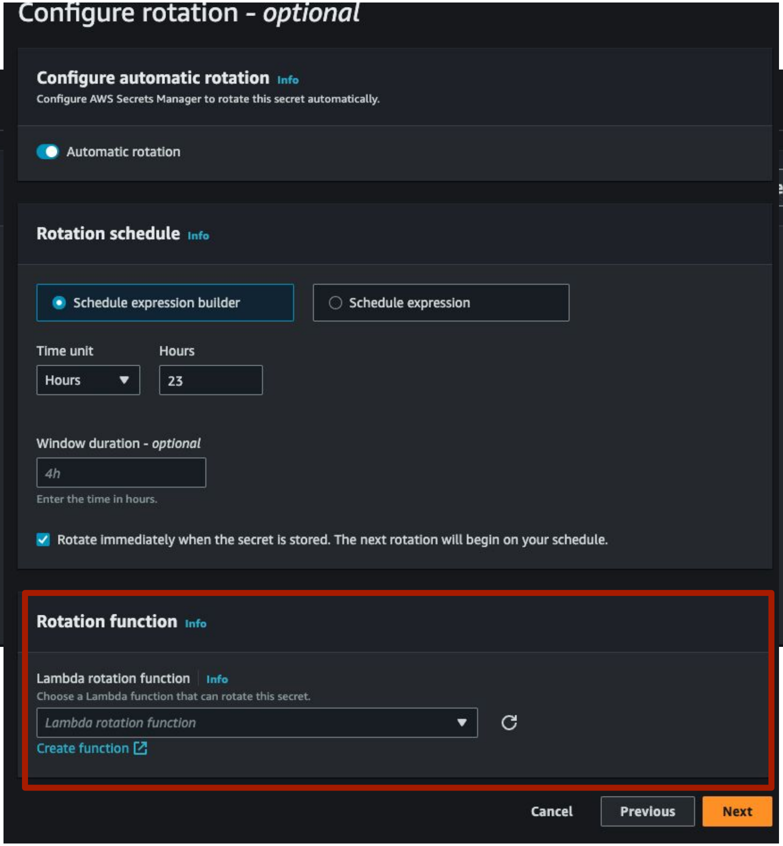
Rotation configuration Info

Rotation status
Enabled

Rotation schedule
7 days

Last rotated date (UTC)
Sun, March 24, 2024 at 06:09:42 UTC

Next rotation date (UTC)
The next rotation is scheduled to occur on or before this date.
Sun, March 31, 2024 at 23:59:59 UTC



Configure rotation - *optional*

Configure automatic rotation Info
Configure AWS Secrets Manager to rotate this secret automatically.

Automatic rotation

Rotation schedule Info

[✓ Schedule expression builder] Schedule expression

Time unit: Hours
Hours: 23

Window duration - *optional*
4h
Enter the time in hours.

✓ Rotate immediately when the secret is stored. The next rotation will begin on your schedule.

Rotation function Info

Lambda rotation function | Info
Choose a Lambda function that can rotate this secret.
Lambda rotation function

Cremit

Additional

KMS vs Secret Manager

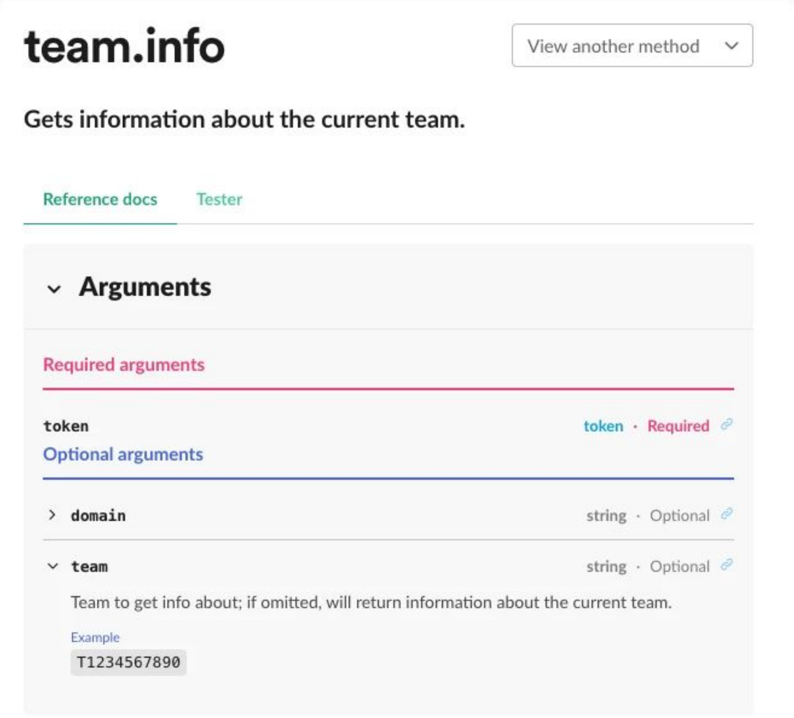
Service	Functionality	Use Case	Key Features
KMS	Encryption & Decryption of “Data”	Protecting Data	안전하고 확장가능한 키 관리, AWS 서비스와의 연동, 쉬운 키 변경
Secret Manager	Storing and managing “Secrets”	Secrets Management	안전하게 저장하고, 관리하는 “Secrets”, 자동 변환, 연동, 관리와 모니터링

Slack Workspace Trick

Cremit

https://hooks.slack.com/services/**T00000000**/B00000000/XXXXXXXXXXXXXXXXXXXXXXXX

https://slack.com/api/team.info?team=**{}**


team.info

View another method

Gets information about the current team.

Reference docs Tester

Arguments

Required arguments

token — token · Required

Optional arguments

domain — string · Optional

team — string · Optional

Team to get info about; if omitted, will return information about the current team.

Example

`T1234567890`