Welcome.html

I

1
2
3
4
5
6
7
8
9
10
11
12
13
14

< Aliyev Omar >

< Haider Zaheer >

< Michael Hernandez >

< Nephthro Pierre >

{

}

http://hawkincloud.github.io/dev/

AWS.html II

{ AWS Shared Responsibility Model }

/01 IAM Users/Groups
/02 IAM Policies/Roles

< Create Users/Groups, Key-id & key-value, Assign group features. Create Policies/Roles, Attach/detach, JSON/Visual editor.

/03 CLI config

< Command line configuration, commands.

/04 Cloudtrail

< Events history, Trails in S3 bucket.

/05 Boto3

< Install Boto3 in Flask, Migrate data events.

}

< Aliyev Omar >

/01 IAM Users/Groups

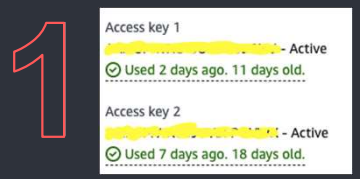
III

IAM = Identity and Access Management, Global service

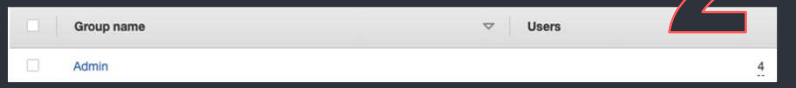
- Root account created by default, shouldn't be used or shared
- Users are people within your organization and can be grouped.
- Groups only contain users, not other groups

IAM = Current HawkinCloud User & Groups

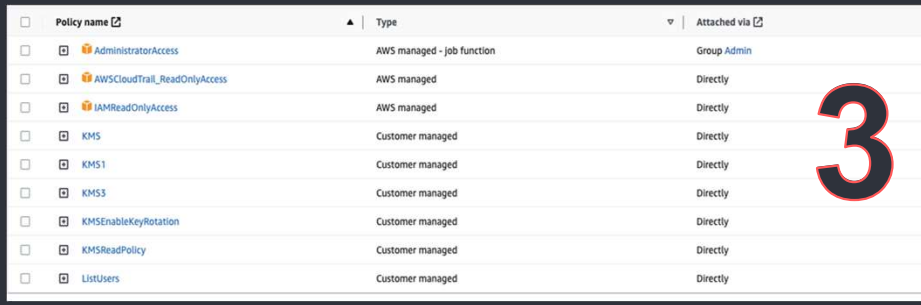
1 • The following access key indicates which user is actively using the configuration in the CLI.



2 • We created an Admin group and assigned each group member a user in the group policy.



3 • It displays the group member "Aliyev" and the policies that are attached to them, creating events in the project based on those policies.



< Aliyev Omar >

/02 IAM Policies/Roles

IV

IAM = Custom JSON & AdminstratorAccess Policies

- Users or Groups can be assigned JSON or given by AWS existing policy documents called policies
- These policies define the permissions of the users

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

The policy provided is a very permissive policy that allows access to all actions on all resources. This policy essentially grants full access to all AWS resources and actions. It is important to note that this policy should be used with caution and only granted to trusted users who require such broad access.

Job Function

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:GenerateCredentialReport",
                "iam:GenerateServiceLastAccessedDetails",
                "iam:Get*",
                "iam:List*",
                "iam:SimulateCustomPolicy",
                "iam:SimulatePrincipalPolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

This policy is more restrictive than the previous one you provided. It grants specific permissions related to the AWS Identity and Access Management (IAM) service. Overall, this policy grants the user permission to view and generate reports about IAM resources, list and get information about IAM resources, and simulate the effects of IAM policies on specified resources. This policy is still quite permissive, but it is more targeted and appropriate for users who require IAM-specific permissions.

Non - Job Function

< Aliyev Omar >

/03 CLI config

V

AWS ACCESS = Management Console, CLI, SDK

- AWS Management Console (protected by password + MFA)
- **AWS Command Line Interface (CLI): protected by access keys**
- **AWS Software Developer Kit (SDK) - for code: protected by access keys**

AWS ACCESS = CLI

- Access Keys are generated through the AWS Console
- Users manage their own access keys

- Access Key ID ~= username
- Secret Access Key ~= password

```
aliyevom@Aliyev-2 ~ % aws configure
AWS Access Key ID [**********ZPNU]:
AWS Secret Access Key [**********9fZZ]:
Default region name [us-east-1]:
Default output format [None]:
aliyevom@Aliyev-2 ~ %
```

AWS ACCESS = CLI include

- Create and manage EC2 instances, load balancers, and other resources
- Create and manage IAM users, groups, roles, and policies
- Create and manage S3 buckets and objects
- Manage your VPC and network resources
- Configure and manage your AWS CLI settings

```
aliyevom@Aliyev-2 ~ % aws iam list-users
{
    "Users": [
        {
            "Path": "/",
            "UserName": "aliyev",
            "UserId": "[redacted]",
            "Arn": "arn:aws:iam::[redacted]:user/aliyev",
            "CreateDate": "2023-02-26T22:00:59+00:00"
        }
    ]
}
```

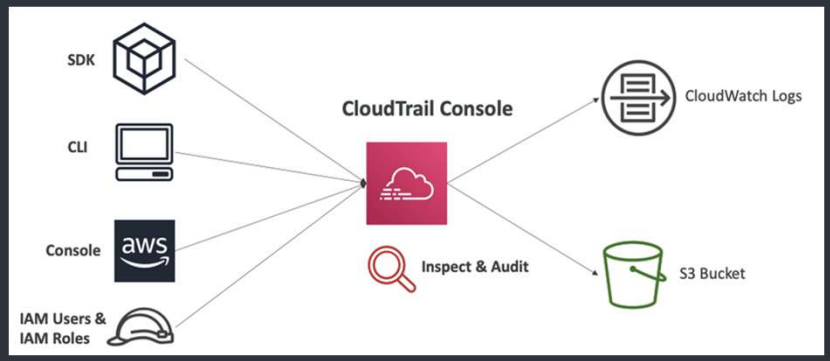
< Aliyev Omar >

/04 Cloudtrail

VI

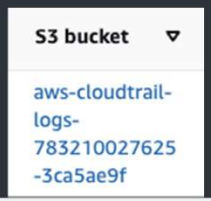
AWS Clodutrail = Provides governance, compliance and audit for your AWS Account

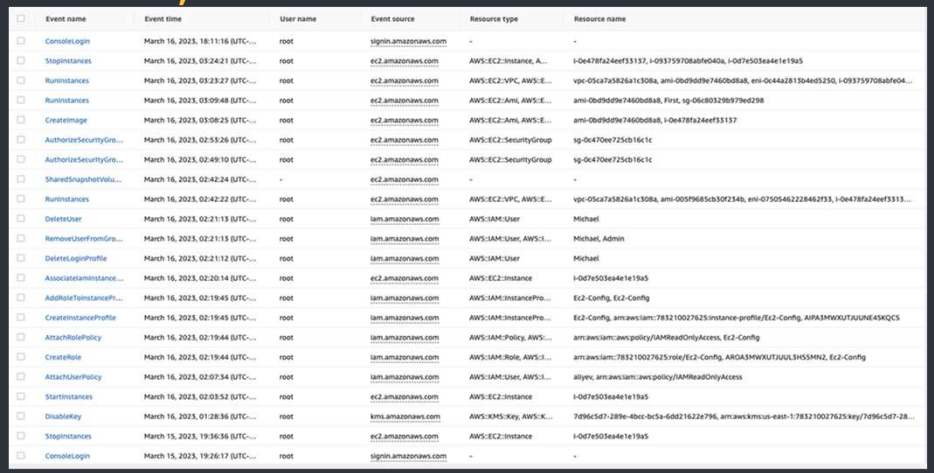
- CloudTrail is enabled by default!
- Get an history of events / API calls made within your AWS Account by:
  - **Console**
  - **SDK**
  - **CLI**
  - **AWS Services**
- Can put logs from CloudTrail into CloudWatch Logs or S3
- A trail can be applied to All Regions (default) or a single Region.
- If a resource is deleted in AWS, investigate CloudTrail first!



AWS Clodutrail = HawkinCloud S3 bucket, History of events, IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudtrail:Get*",
                "cloudtrail:Describe*",
                "cloudtrail:List*",
                "cloudtrail:LookupEvents"
            ],
            "Resource": "*"
        }
    ]
}
```



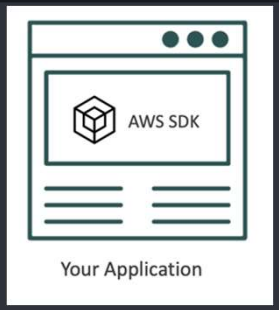


< Aliyev Omar >

/05 Boto3

AWS Software Development Kit = AWS SDK

- Enables you to access and manage AWS services programmatically
- Supports:
  - SDKs (JavaScript, Python, PHP, .NET, Ruby, Java, Go, Node.js, C++)
  - Mobile SDKs (Android, iOS, …)
  - IoT Device SDKs (Embedded C, Arduino, …)



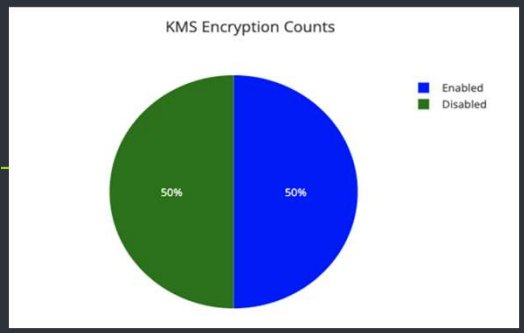
AWS CLI is built on AWS SDK for Python = HawkinCloud

```python
client = boto3.client('cloudtrail')
# List all events for a specific trail
# List all trails
response = client.lookup_events()
print(response)
events=[]
for event in response['Events']:
    events.append({
        'event_id': event['EventId'],
        'event_name': event['EventName'],
        'event_time': event['EventTime'],
        'event_detail': event['CloudTrailEvent'],
    })
    print("Event: ", event['EventId'])
# Render the template with the events data
print(events)
return render_template('dashboard.html', name=current_user.username, events=events)
```

```python
@app.route('/kms_data')
def kms_data():
    keys = ['Enabled', 'Disabled']
    enabled_count = 0
    disabled_count = 0
    client = boto3.client('kms')
    keys_list = client.list_keys()
    for key in keys_list["Keys"]:
        rotation_response = client.get_key_rotation_status(
            KeyId=str(key["KeyId"])
        )
        if(rotation_response["KeyRotationEnabled"] is True):
            enabled_count+=1
        else:
            disabled_count+=1
    rotation_count = [enabled_count, disabled_count]

    return jsonify({
        'keys': keys,
        'rotation_count': rotation_count,
    })
```

Event ID	Event Name	Event Time	Event Detail
f682edc7-4c47-4b5a-bd5e-4b1cbce994d7	DescribeEventAggregates	2023-03-16 20:14:51-04:00	{"eventVersion":"1.08","userIdentity":{"type":"Root","principalId":"783210027625","arn":"arn:aws:iam::783210027625:root","accountId":"783210027625","accessKeyId":"ASIA3MWXUTJUQNU52P3E","sessionContext":{"sessionIssuer":{},"webIdFederationData":{},"attributes":{"creationDate":"2023-03-16T22:11:16Z","mfaAuthenticated":"false"}}},"eventTime":"2023-03-17T00:14:51Z","eventSource":"health.amazonaws.com","eventName":"DescribeEventAggregates","awsRegion":"us-east-1","sourceIPAddress":"66.31.54.133","userAgent":"AWS Internal","requestParameters":{"aggregateField":"eventTypeCategory","filter":{"eventStatusCodes":["open","upcoming"],"startTimes":[{"from":"Mar 10, 2023 12:14:51 AM"}]}},"responseElements":null,"requestID":"7683d68a-6dbf-4653-99ad-b0e3eb689691","eventID":"f682edc7-4c47-4b5a-bd5e-4b1cbce994d7","readOnly":true,"eventType":"AwsApiCall","managementEvent":true,"recipientAccountId":"783210027625","eventCategory":"Management","sessionCredentialFromCor



HawkinCloud Status

Since the most recent day of class—presentation day, our group has made substantial efforts in working on both the frontend and backend.

With almost 2 months into the initiation of the AWS project, HawkinCloud, we have made drastic changes on the front-end, with the creation displaying significant metrics as they relate to the AWS Cloudtrail as displayed by dashboard.html—integrated with Flask.

We continue to meet more than once a week, either virtually or in person. Significant progress on both the frontend and backend have fast-tracked HawkinCloud.

HawkinCloud Timeline (Recap)

Phase I - Conceptualization
As discussed in our introductory presentation

Phase II - Initialization
Frontend (HTML), Backend (Python), and Integration (Flask) are connected and arranged for primary functionality

Phase III - Optimization
CloudTrail/Boto3 are implemented for HawkInCloud's full AWS services

End.html

X

}

http://hawkincloud.github.io/dev/