WeCloudData

Introduction to AWS

Prepared by WeCloudData

Cloud
AWS Introduction

- **AWS Services**
- User and IAM Role
- Compute: EC2
- Storage: S3
- AWS CLI and APIs

Agenda.

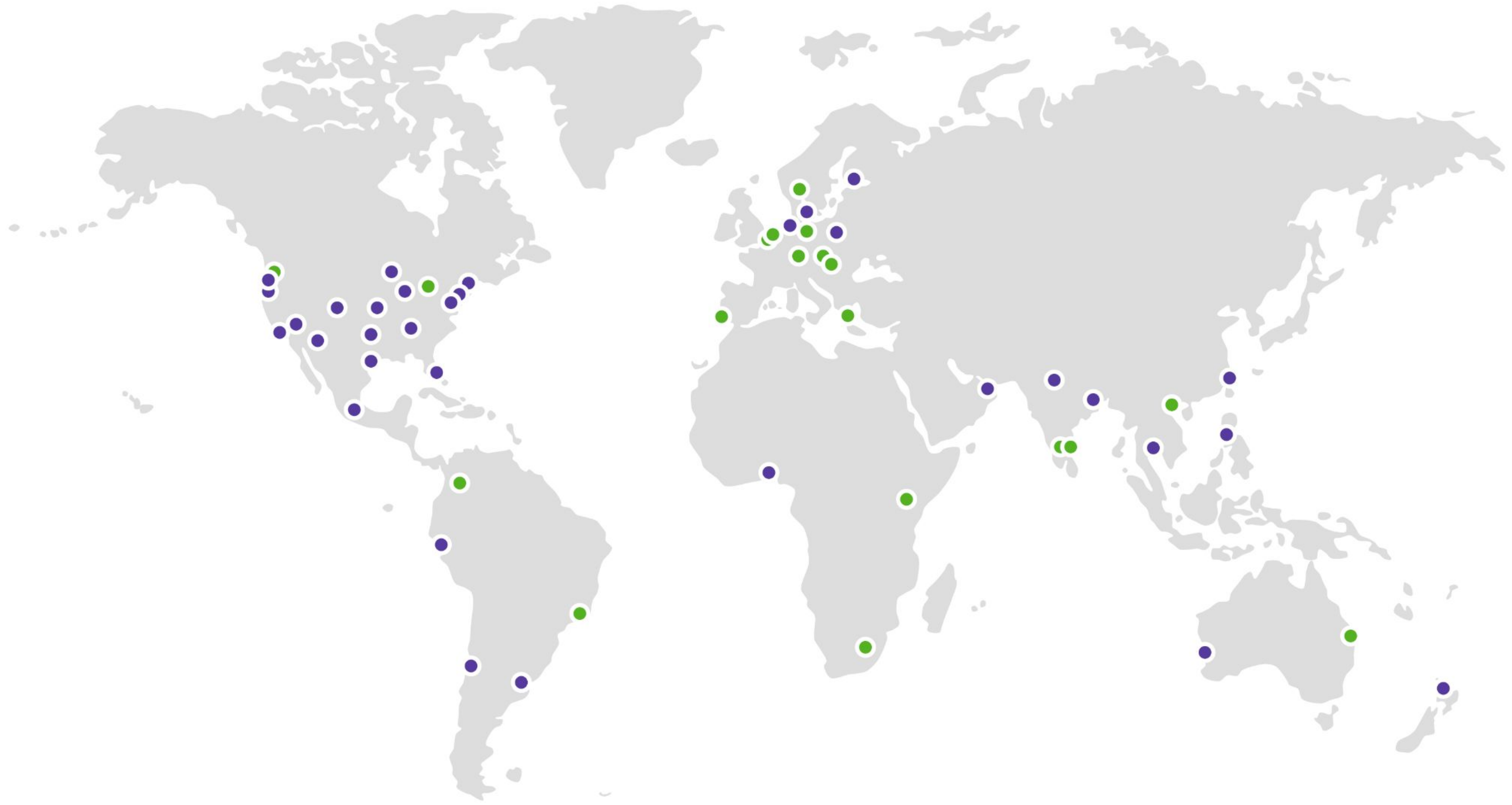
Data Centers

AWS Intro

Data Centers

Global Regions & Availability Zones

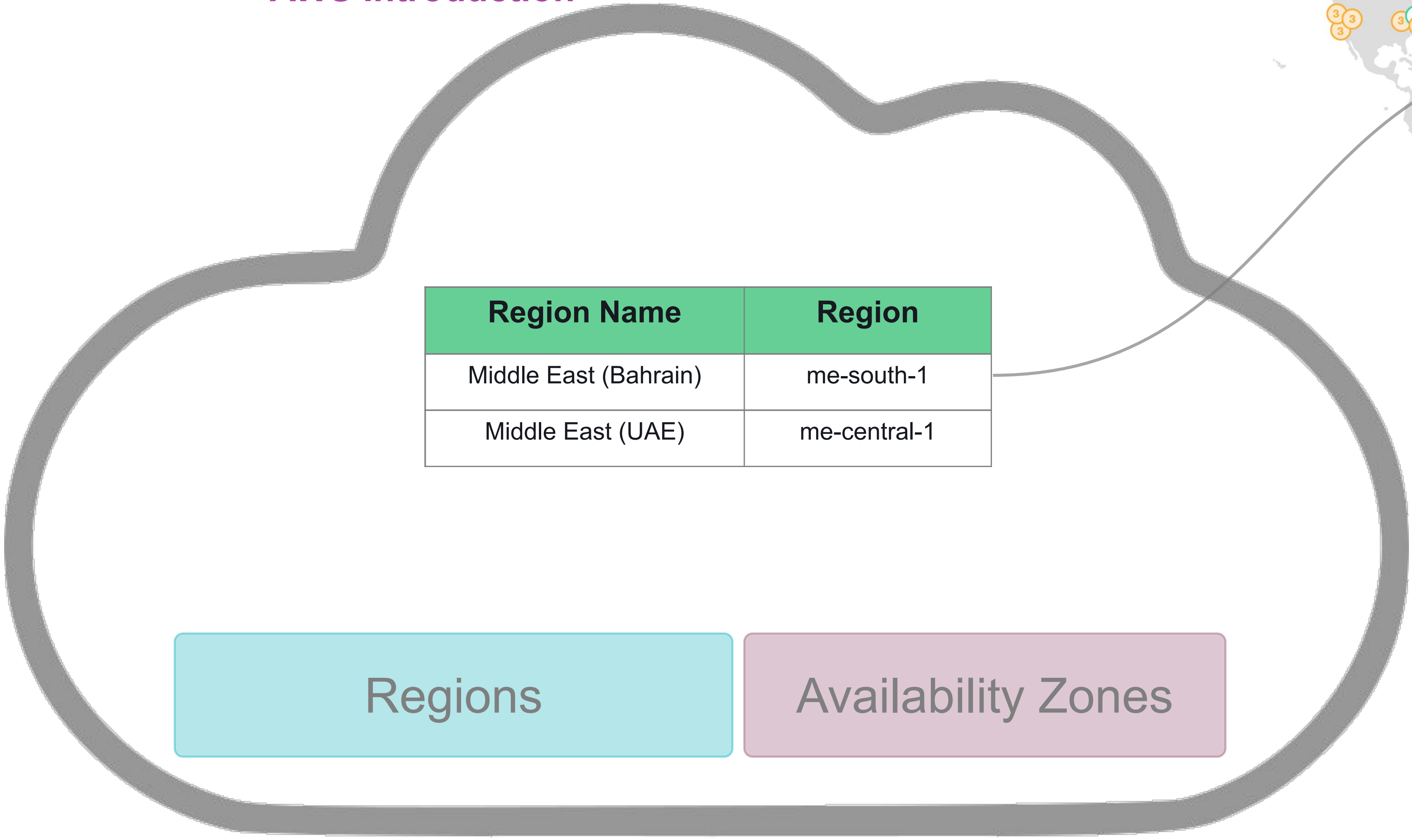
AWS Introduction



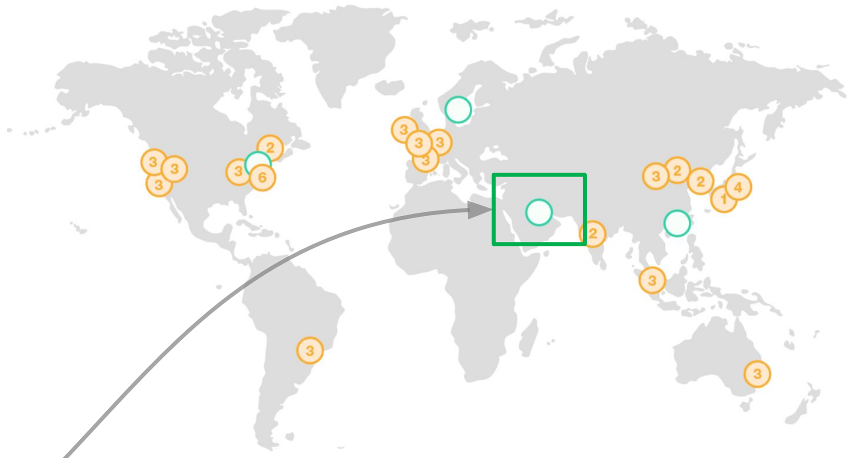
WeCloudData

Regions and Availability Zones

AWS Introduction



Region Name	Region
Middle East (Bahrain)	me-south-1
Middle East (UAE)	me-central-1

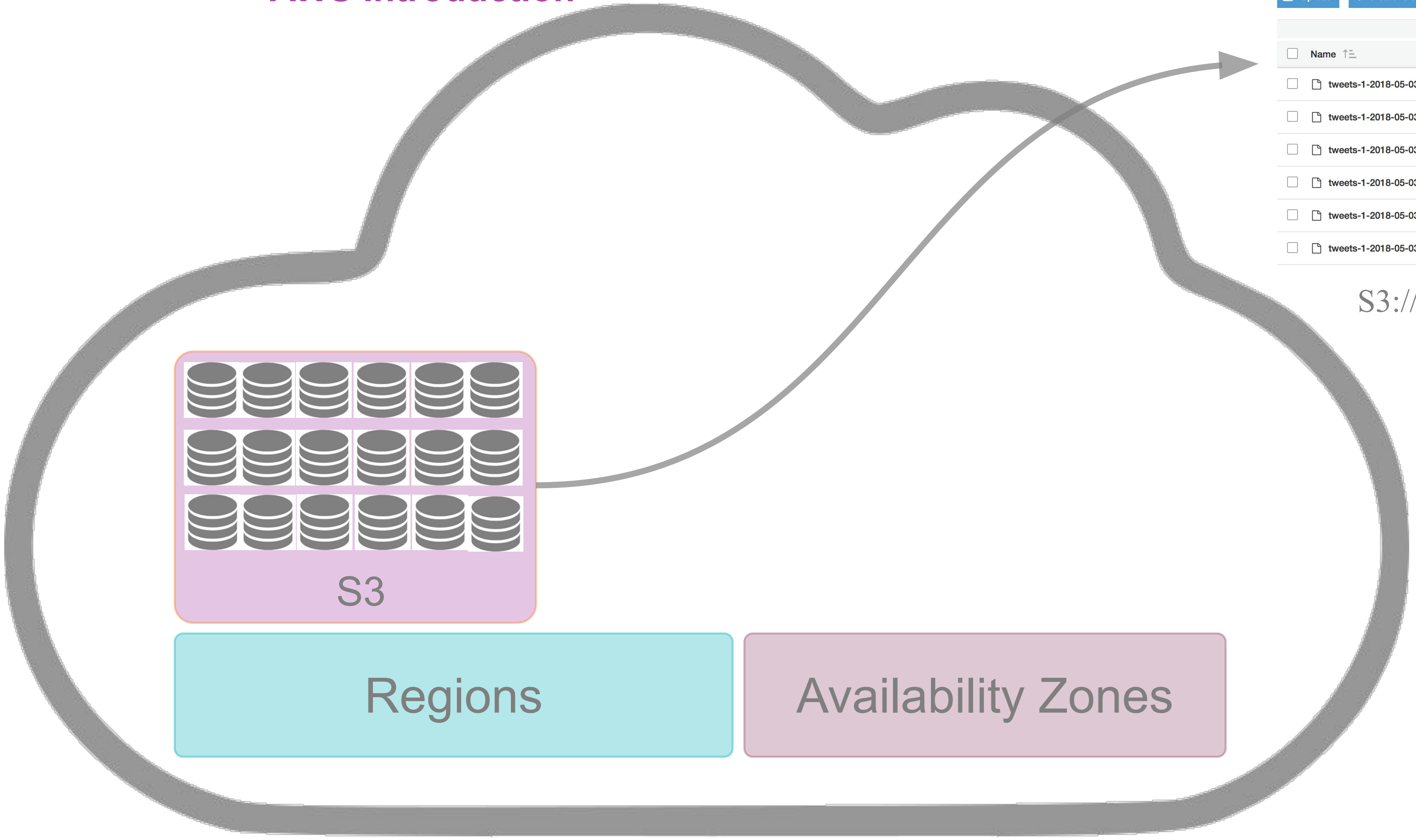


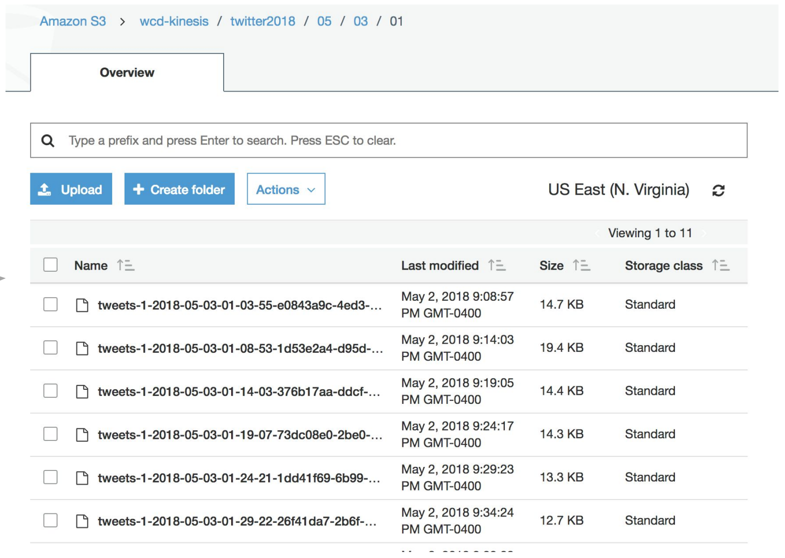
Region	AZ
virginia	us-east-1a us-east-1b us-east-1c
oregon	us-west-2a us-west-2b us-west-2c
canada	ca-central-1a ca-central-1b

WeCloudData

Storage: S3 Object Store

AWS Introduction




Amazon S3 > wcd-kinesis / twitter2018 / 05 / 03 / 01

Overview

Type a prefix and press Enter to search. Press ESC to clear.

Upload | Create folder | Actions | US East (N. Virginia)

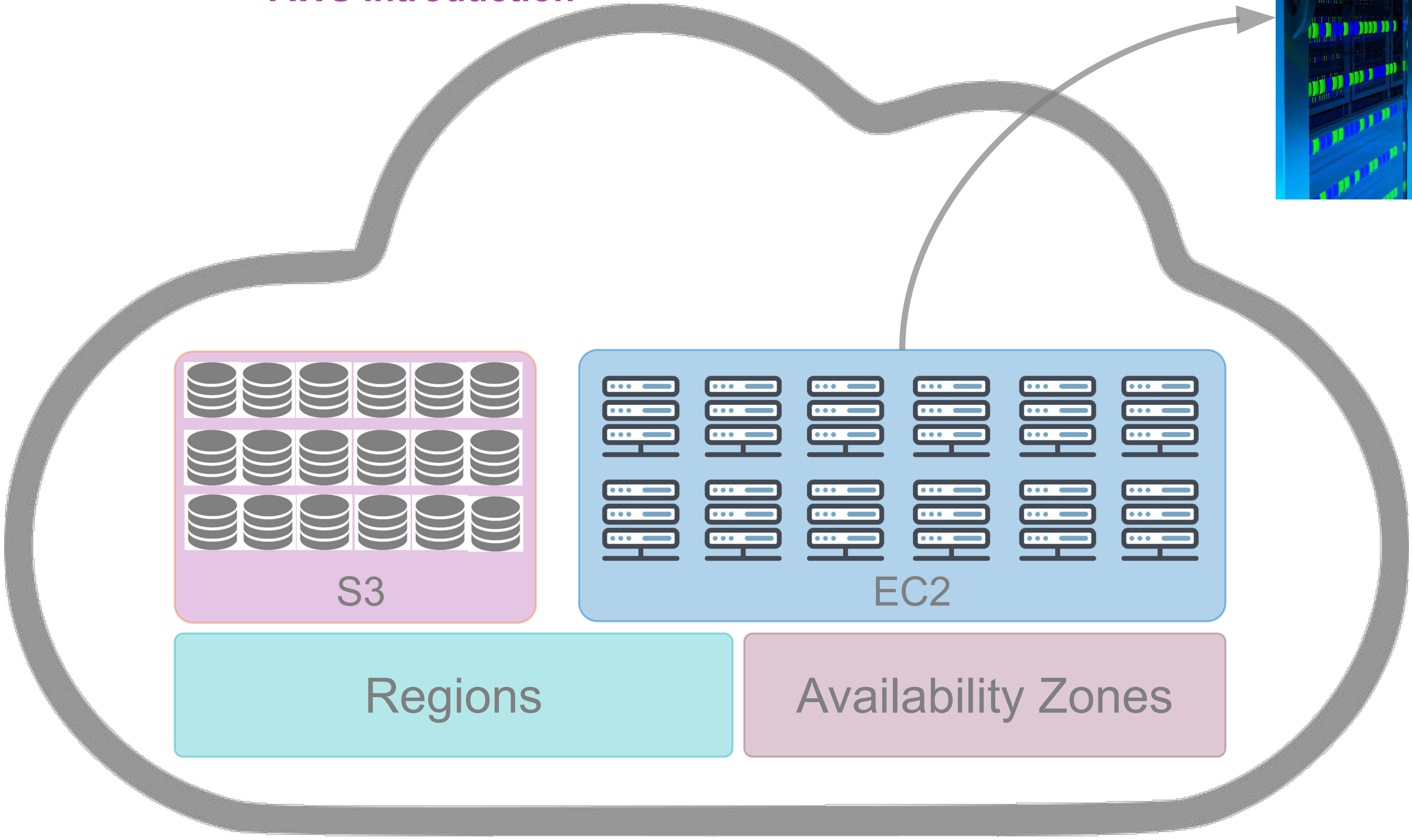
Viewing 1 to 11

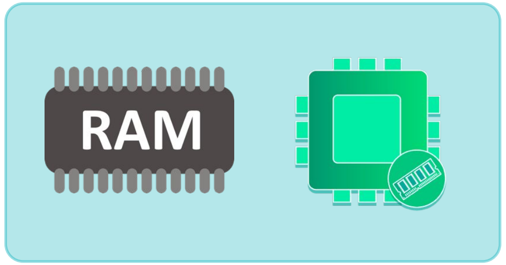
Name	Last modified	Size	Storage class
tweets-1-2018-05-03-01-03-55-e0843a9c-4ed3-…	May 2, 2018 9:08:57 PM GMT-0400	14.7 KB	Standard
tweets-1-2018-05-03-01-08-53-1d53e2a4-d95d-…	May 2, 2018 9:14:03 PM GMT-0400	19.4 KB	Standard
tweets-1-2018-05-03-01-14-03-376b17aa-ddcf-…	May 2, 2018 9:19:05 PM GMT-0400	14.4 KB	Standard
tweets-1-2018-05-03-01-19-07-73dc08e0-2be0-…	May 2, 2018 9:24:17 PM GMT-0400	14.3 KB	Standard
tweets-1-2018-05-03-01-24-21-1dd41f69-6b99-…	May 2, 2018 9:29:23 PM GMT-0400	13.3 KB	Standard
tweets-1-2018-05-03-01-29-22-26f41da7-2b6f-…	May 2, 2018 9:34:24 PM GMT-0400	12.7 KB	Standard

S3://bucket/folder/object

EC2: Compute

AWS Introduction

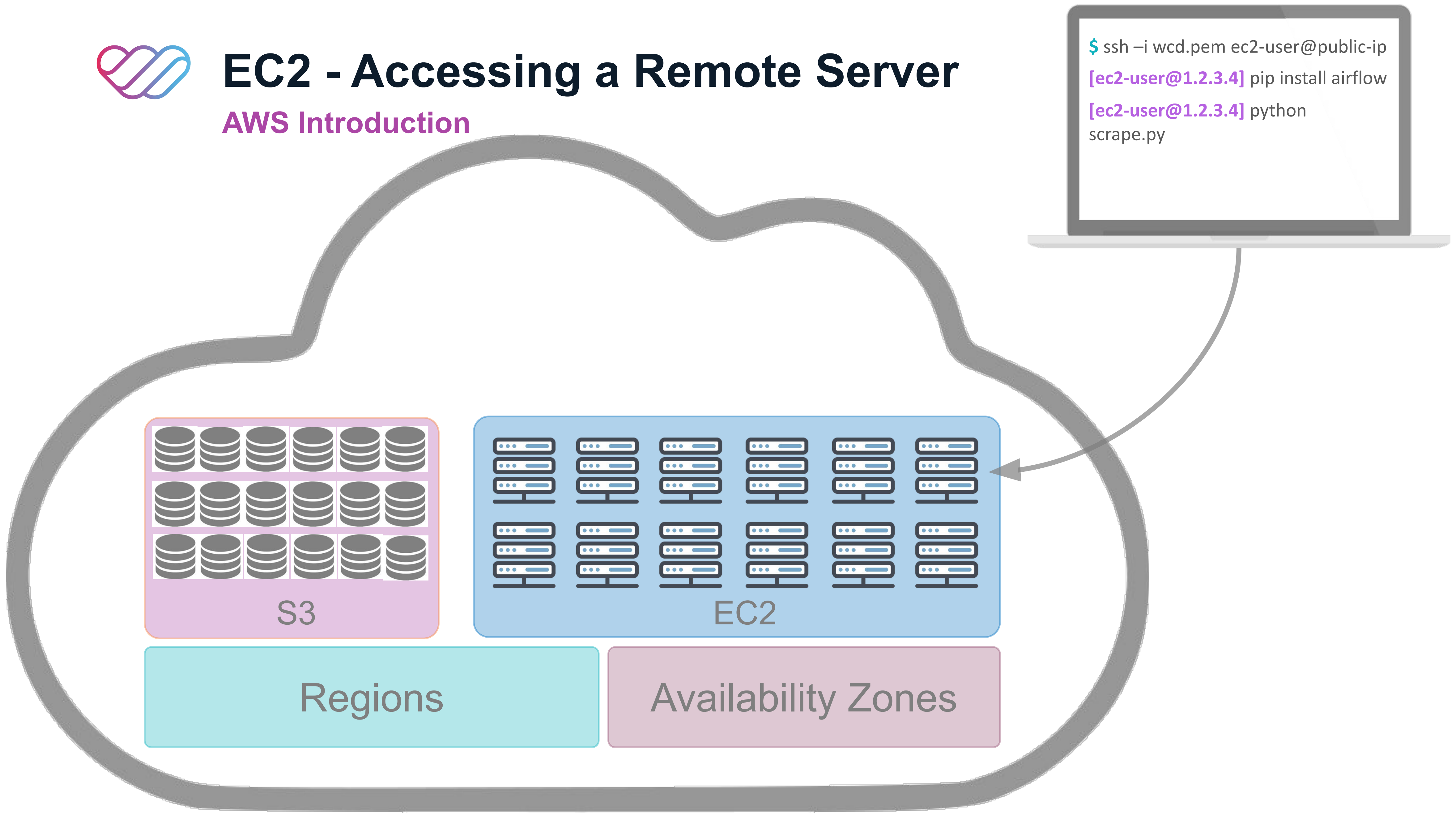




WeCloudData

EC2 - Accessing a Remote Server

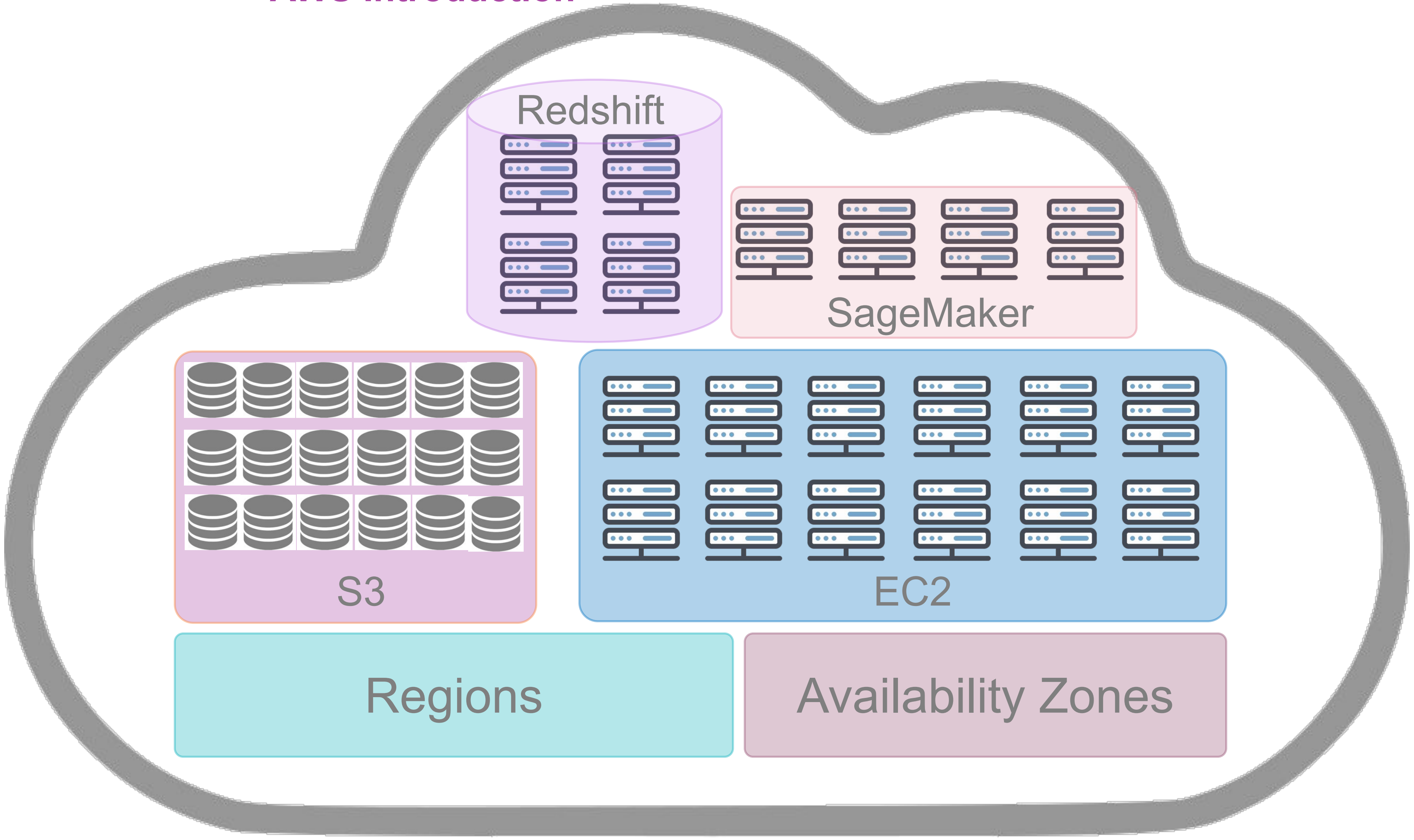
AWS Introduction



```
$ ssh –i wcd.pem ec2-user@public-ip
[ec2-user@1.2.3.4] pip install airflow
[ec2-user@1.2.3.4] python
scrape.py
```

EC2 - Accessing a Remote Server

AWS Introduction

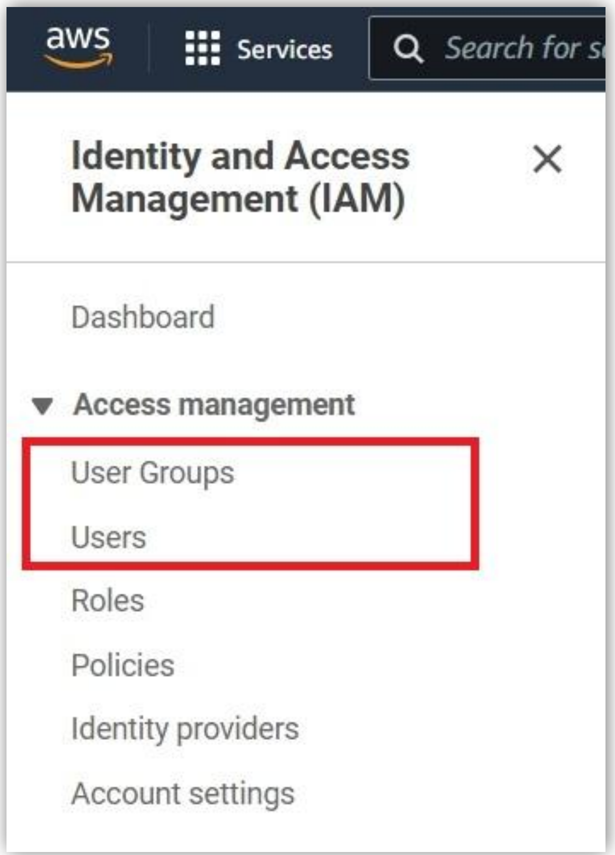


AWS
User Accounts & Roles

- AWS Services
- **User and IAM Role**
- Compute: EC2
- Storage: S3
- AWS CLI and APIs

Agenda.

AWS IAM – Concepts



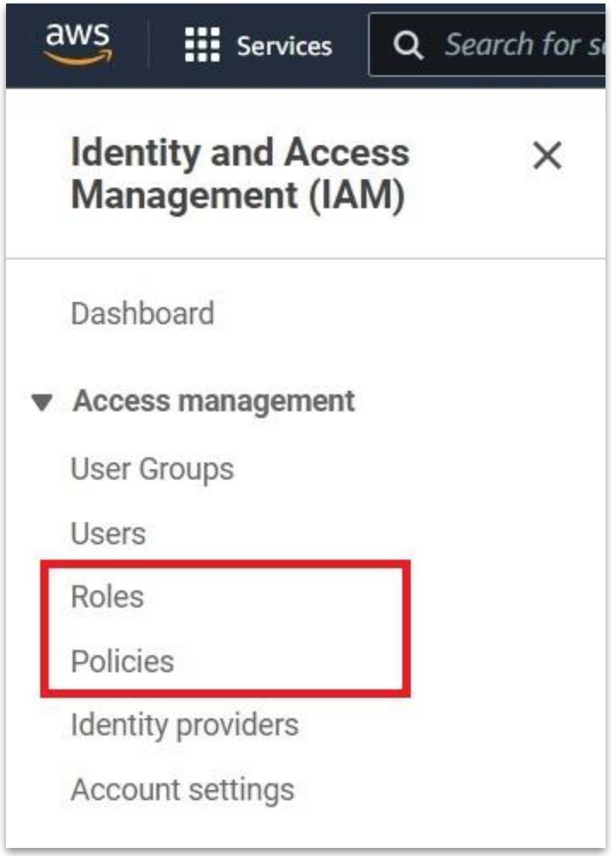
User

A IAM user is a **person** that needs to interact with your AWS resources or services either from the AWS Console or with the AWS CLI. When you create a new user, no credentials are assigned, and the user does not have any permission to access your AWS resources.

User Group

An IAM group is a **collection of users and permissions** assigned to those users. Groups provide a convenient way to manage permissions for users with similar needs by categorizing them according to their job function/role, department, or any other requirement. Then, permissions for all those users can be managed at once through the group.

AWS IAM – Concepts



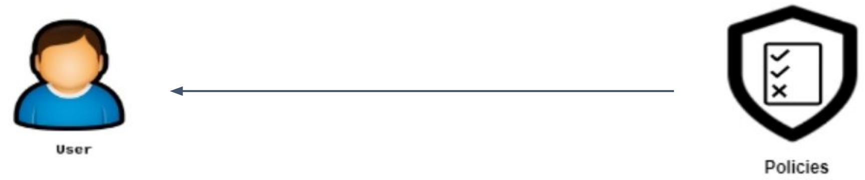
Role

An IAM role is an **entity** within AWS which defines **a set of permissions the role can perform**, and what entities can assume the role. **A role is not directly linked to a person or a service**, rather it can be assumed by any resource that the role grants permission to. Additionally, Roles allow you to grant multi-account access to your AWS resources from users, services, and apps that aren't part of your business.

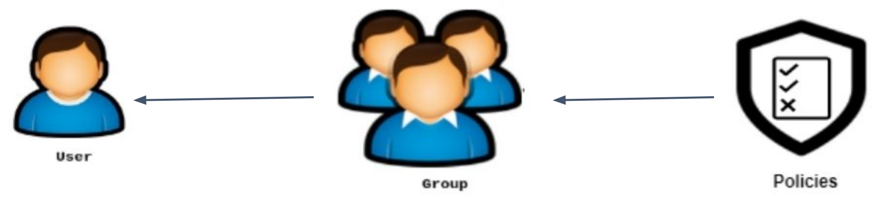
Permission Policy

A policy is a document with **a set of rules**, having one or more statements. Each policy grants a specific set of permissions and can be attached to any of the IAM identities we covered earlier—users, groups, and roles. Policies are always written in JSON or YAML format and each policy has a name.

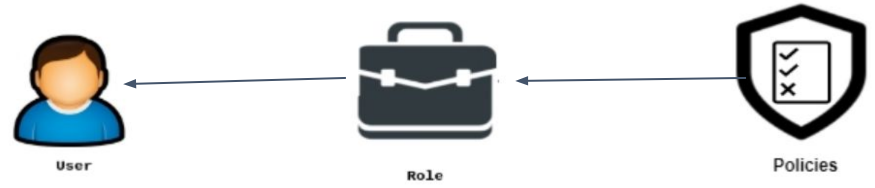
AWS IAM – Relationship



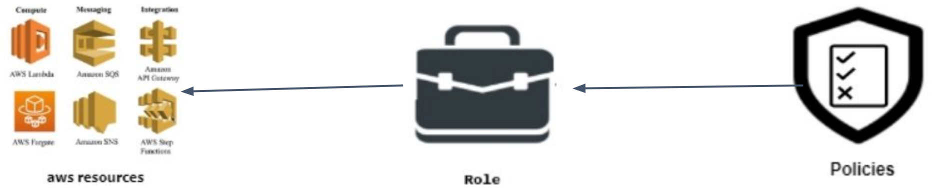
Policy–User:
Policies are assigned to the single user.



Policy–User Group – User:
Policies are assigned to a user group, and the a user group is assigned to a user. In this way, if the permissions are defined for a group, it will defined for the users.



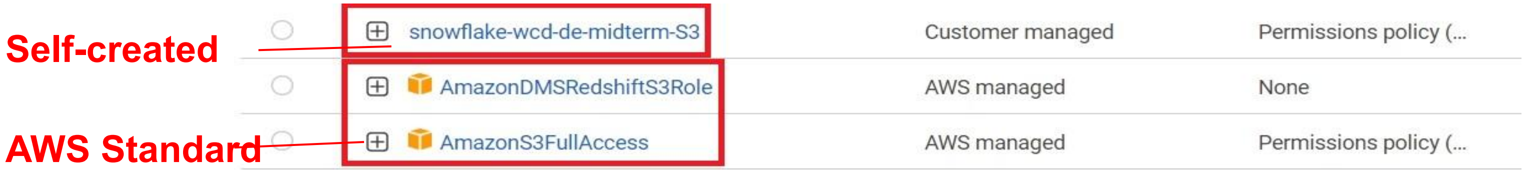
Policy–Role – User:
Policies are assigned to a Role, then the a role is assigned to a user. In this way, if the permissions are defined for a role, it will defined for the user.



Policy–Role – Service:
Policies are assigned to a Role, then the a role is assigned to a AWS service. In this way, an AWS service have access to other AWS resources.

AWS IAM – Policy

A policy is an object in AWS that defines permissions. It a JSON document which **allows or denies one or many permissions** on a principal. A principal can be a role, a user, or a group. Since everything is **denied by default**, the most common case is that **a policy will allow an action**. You can use the standard AWS policies, or create your own policy.



⊞ snowflake-wcd-de-midterm-S3	Customer managed	Permissions policy (...
⊞ AmazonDMSRedshiftS3Role	AWS managed	None
⊞ AmazonS3FullAccess	AWS managed	Permissions policy (...

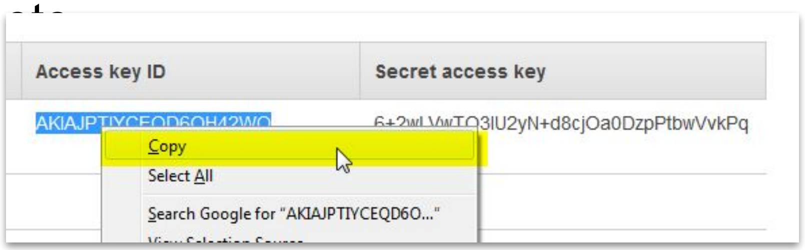
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*",
                "s3-object-lambda:Get*",
                "s3-object-lambda:List*"
            ],
            "Resource": "arn:aws:s3:::example_bucket"
        }
    ]
}
```

A Policy Example:

- **Effect:** Always "Allow", because for all resources, default access is **Deny**.
- **Action**: What work can do under this policy.
- **Resource**: What resource the policy allow to access. In this example, the "example_bucket" is the only resource that is allowed to access. ("arn" is an unique aws id for various entity.)

AWS IAM – User

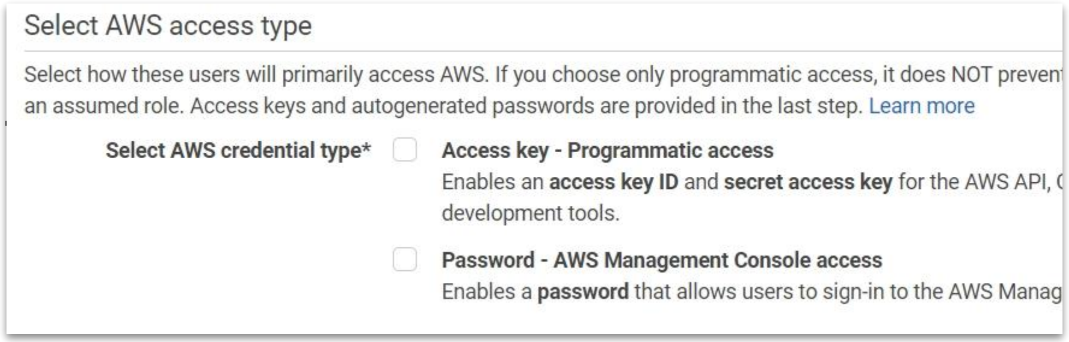
- There are two different types of users in AWS. You are either the account owner (**root user**) or you are an **IAM user**. The root user is created when the AWS account is created. IAM users are created by the root user. All AWS users have **security credentials**.
- **security credentials**: is a Access_Key/Secret_access_key pair that is used to make programmatic calls to AWS from many places, like calls from linux, from python script, etc.



Access key ID	Secret access key
AKIAJPTIYCEQD6OH42WO	6+2wLVwTQ3lU2yN+d8cjOa0DzpPtbwVvkPq

- For a user, there are 2 access type:
  - access from programmatic call, like CLI or API.
  - access from the AWS Console

You will choose the type when you create a user.



Select AWS access type

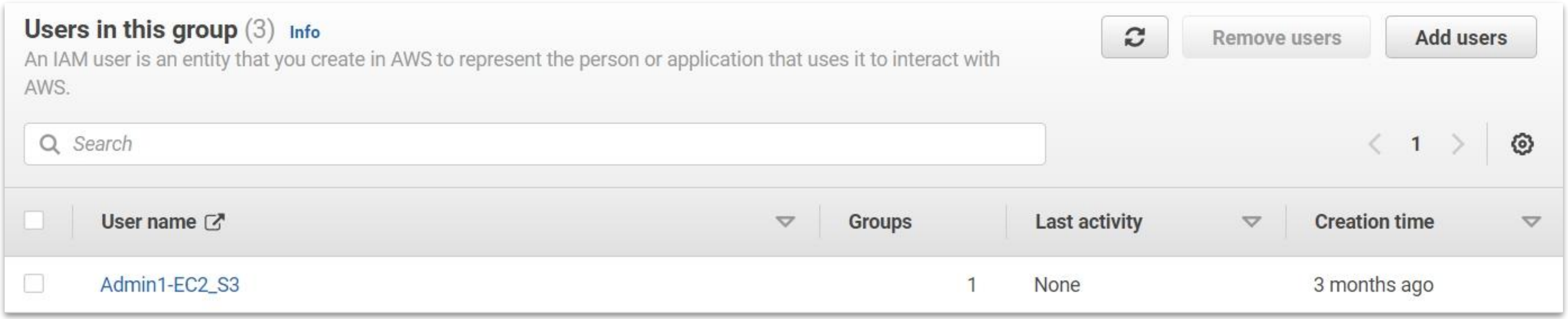
Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent an assumed role. Access keys and autogenerated passwords are provided in the last step. Learn more

Select AWS credential type*

- [ ] Access key - Programmatic access
  Enables an **access key ID** and **secret access key** for the AWS API, development tools.
- [ ] Password - AWS Management Console access
  Enables a **password** that allows users to sign-in to the AWS Manag

AWS IAM – User Group

- A user group is a collection of IAM users. Use groups to specify permissions for a collection of users.



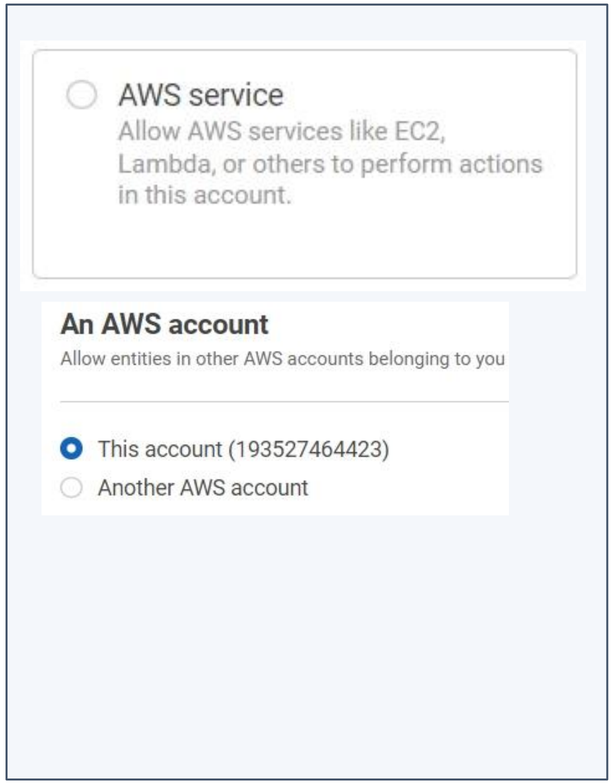
Users in this group (3) Info

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

User name	Groups	Last activity	Creation time
Admin1-EC2_S3	1	None	3 months ago

AWS IAM – Role

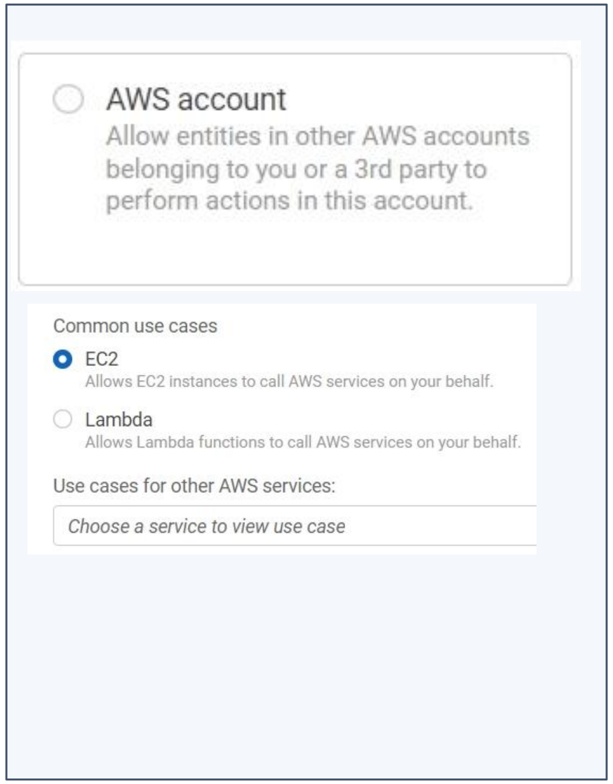
- An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations.
- A Role can be created by various entity types:



AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

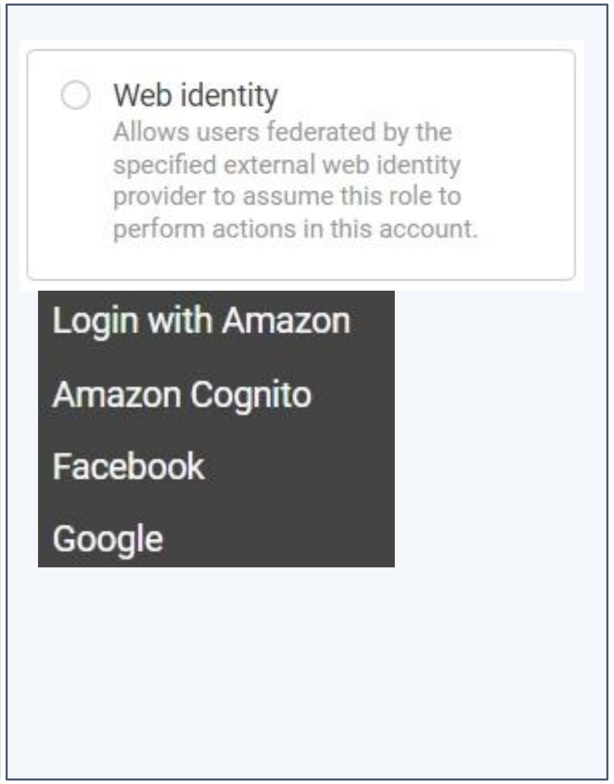
An AWS account
Allow entities in other AWS accounts belonging to you

[✓ This account (193527464423)]
Another AWS account



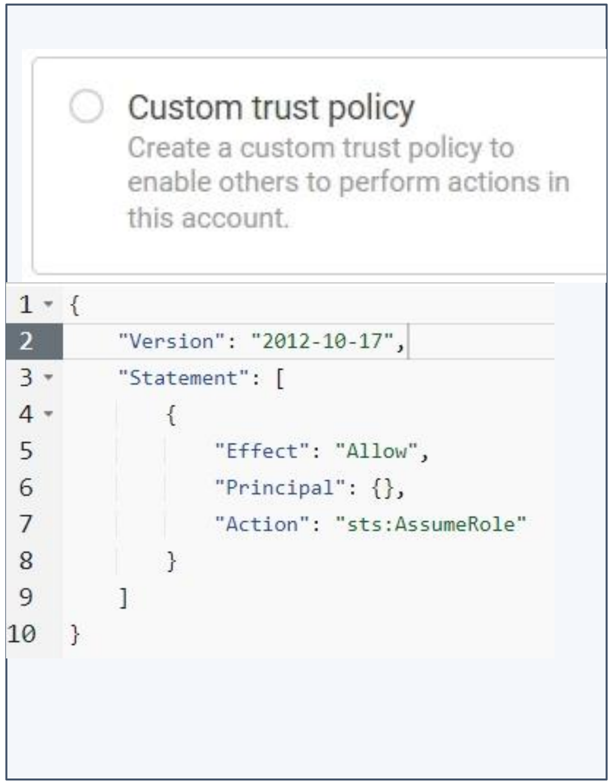
AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

Common use cases
[✓ EC2]
Allows EC2 instances to call AWS services on your behalf.
Lambda
Allows Lambda functions to call AWS services on your behalf.
Use cases for other AWS services:
Choose a service to view use case



Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

Login with Amazon
Amazon Cognito
Facebook
Google



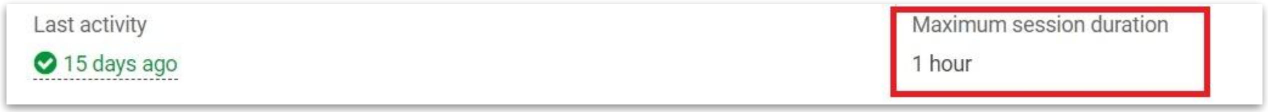
Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {},
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Try in system

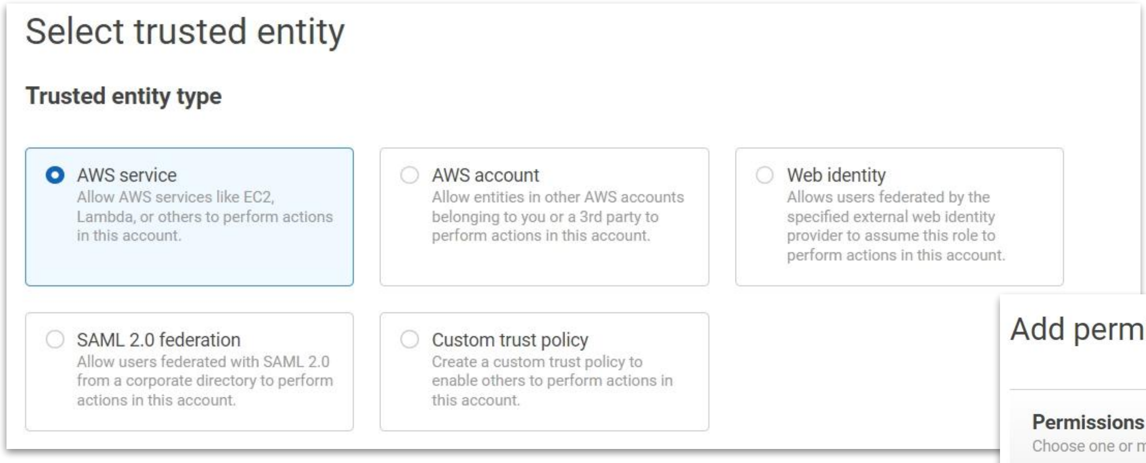
AWS IAM – Role

- An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations.



Last activity	Maximum session duration
15 days ago	1 hour

- An IAM role can be assigned to an identity, such as user or user group, or an aws service, such an EC2 instance.



Select trusted entity

Trusted entity type

[✓ AWS service] Allow AWS services like EC2, Lambda, or others to perform actions in this account.

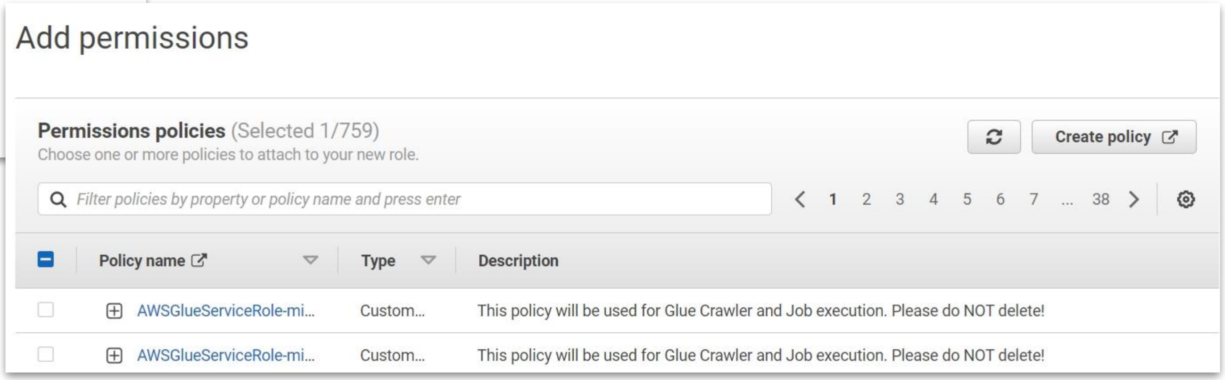
AWS account — Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

Web identity — Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

SAML 2.0 federation — Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

Custom trust policy — Create a custom trust policy to enable others to perform actions in this account.

- Policies will be attached to the Role.



Add permissions

Permissions policies (Selected 1/759)
Choose one or more policies to attach to your new role.

Filter policies by property or policy name and press enter

	Policy name	Type	Description
	AWSGlueServiceRole-mi...	Custom...	This policy will be used for Glue Crawler and Job execution. Please do NOT delete!
	AWSGlueServiceRole-mi...	Custom...	This policy will be used for Glue Crawler and Job execution. Please do NOT delete!

WeCloudData

Follow-along

Follow-along

User Account & Roles

Follow

DEMO: follow-along

Sign in to AWS

Explore a list of AWS services

AWS
EC2 (Elastic Compute)

AWS Services

User and IAM Role

Compute: EC2

Storage: S3

AWS CLI and APIs

Agenda.

EC2

AWS Compute

What is EC2?

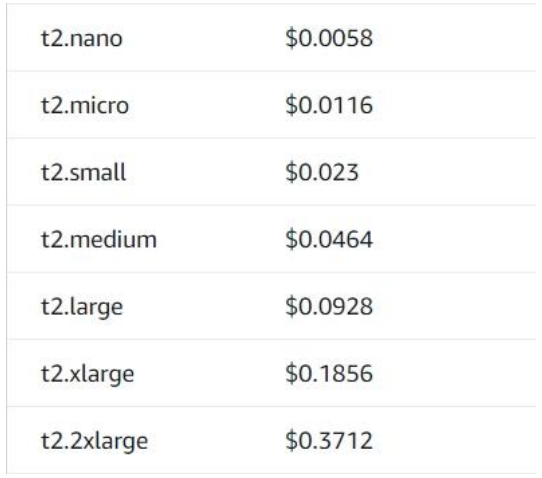
EC2(Elastic Cloud Compute) a cloud server. What you're actually getting with EC2 is an on demand virtual machine which AWS calls an "instance."

Why use EC2?

- Use EC2 as servers to install softwares in the data project, like Docker or Apache Airbyte.
- Use EC2 as servers to run some scheduled jobs, like python and shell scripts.
- Use EC2 as clusters for Apache Spark.

EC2 Pricing

- For new users (registration <12 months), AWS provides Free Tier including 750 hours of Linux and Windows t2.micro instances, (t3.micro for the regions in which t2.micro is unavailable) each month for one year. To stay within the Free Tier, use only EC2 Micro instances. This is the price example for t2. us-east.



t2.nano	$0.0058
t2.micro	$0.0116
t2.small	$0.023
t2.medium	$0.0464
t2.large	$0.0928
t2.xlarge	$0.1856
t2.2xlarge	$0.3712

Useful link: https://aws.amazon.com/ec2/pricing/on-demand/

EC2 Lab

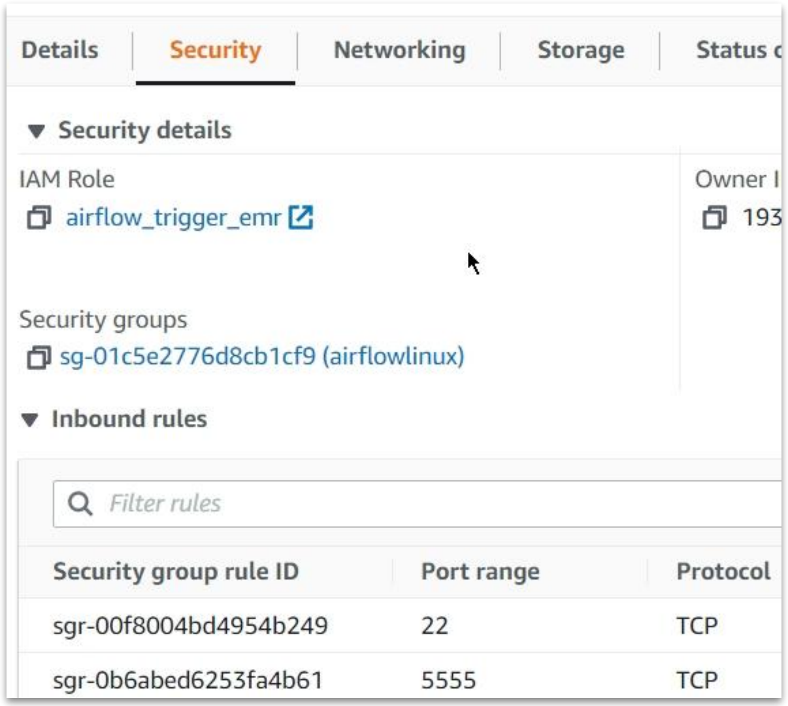
Lab

Demo

A running EC2 instance

EC2

AWS Compute



Details | Security | Networking | Storage | Status c

▼ Security details

IAM Role
airflow_trigger_emr

Owner I
193

Security groups
sg-01c5e2776d8cb1cf9 (airflowlinux)

▼ Inbound rules

Filter rules

Security group rule ID	Port range	Protocol
sgr-00f8004bd4954b249	22	TCP
sgr-0b6abed6253fa4b61	5555	TCP

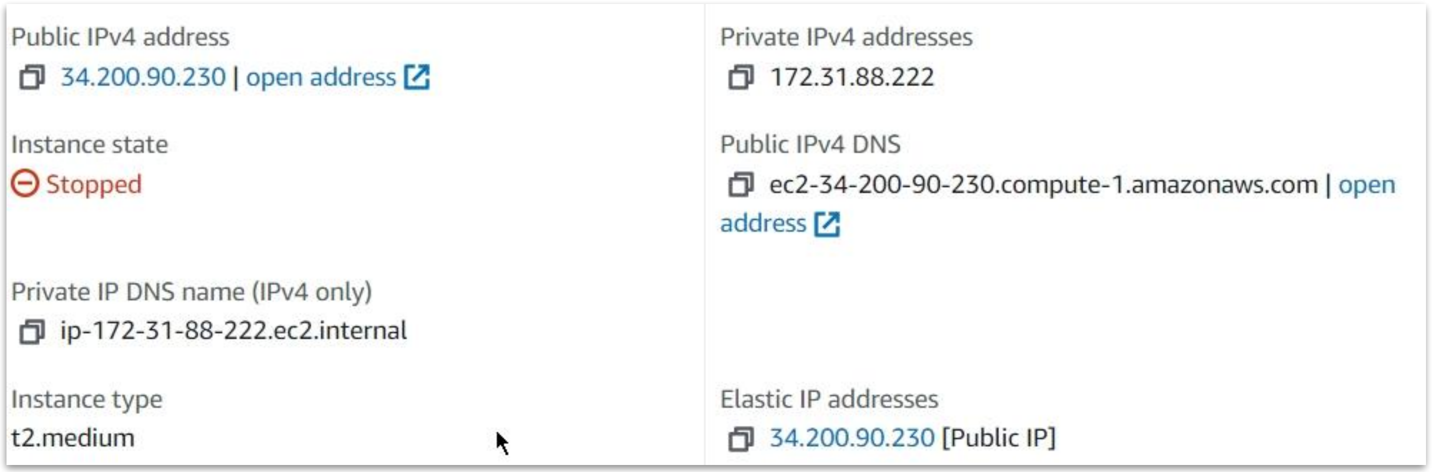
Security Group

A security group acts as a virtual firewall, controlling the traffic that is allowed to reach and leave the resources that it is associated with. For example, after you **associate a security group with an EC2 instance**, it controls the **inbound and outbound traffic** for the instance.

Try in system

EC2

AWS Compute



Public IPv4 address
34.200.90.230 | open address

Private IPv4 addresses
172.31.88.222

Instance state
Stopped

Public IPv4 DNS
ec2-34-200-90-230.compute-1.amazonaws.com | open address

Private IP DNS name (IPv4 only)
ip-172-31-88-222.ec2.internal

Instance type
t2.medium

Elastic IP addresses
34.200.90.230 [Public IP]

Public IPv4 address

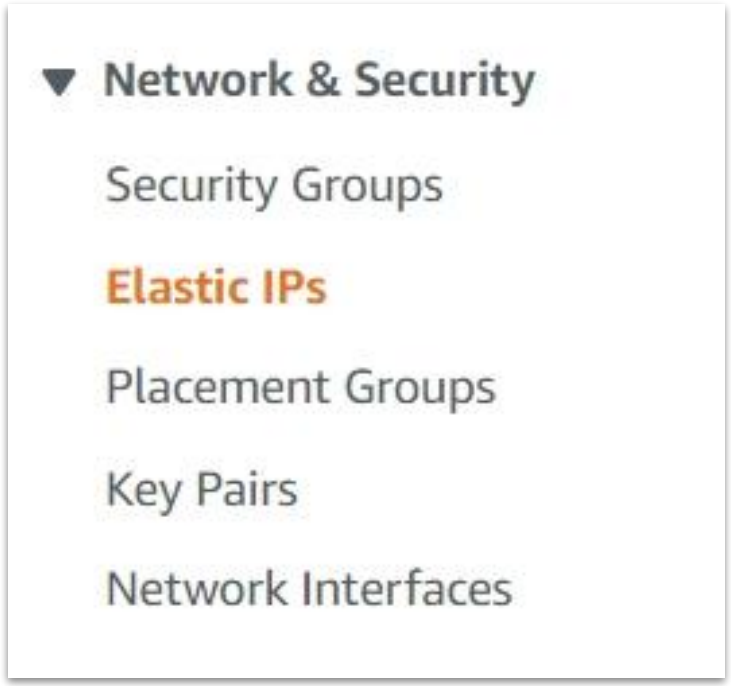
The public IPv4 address is an IP address that can be **accessed directly over the internet** and is assigned to your network router by your internet service provider. If you installed application on your EC2 instance, the IPv4 address will be the host address for your application.

Be aware that the IPv4 address of an EC2 instance is dynamic, which means that every time when you restart your instance, the IP address is different. In order to have a stable address you need to assign an **Elastic IP** to the EC2 instance.

Try in system

EC2

AWS Compute



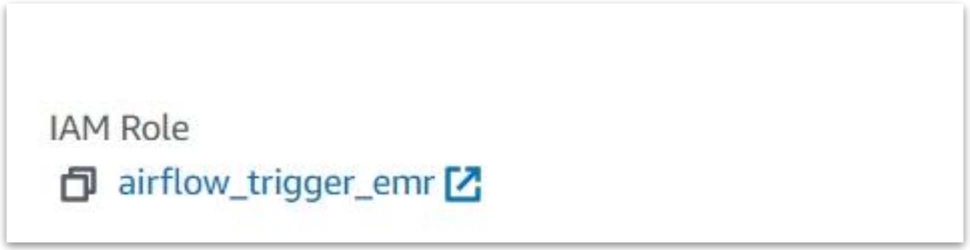
Elastic IPs

An Elastic IP address is a static IPv4 address designed for dynamic cloud computing. By using an Elastic IP address, you can mask the failure of an instance or software by rapidly remapping the address to another instance in your account.

Try in system

EC2

AWS Compute


IAM Role
airflow_trigger_emr

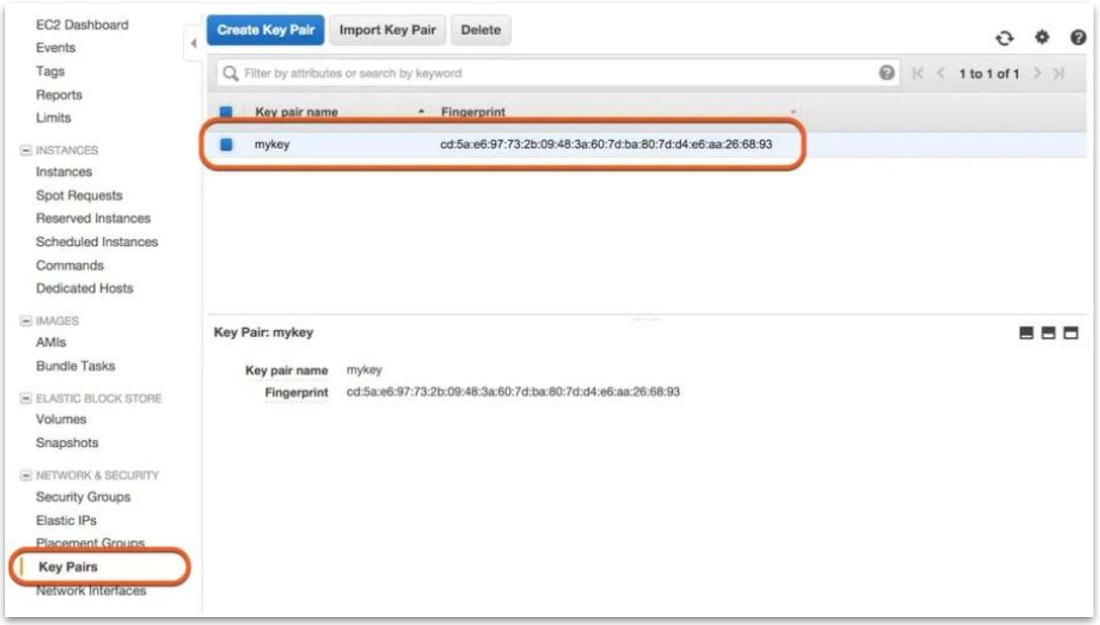
IAM Role
As we discussed in previous pages, when we assign an IAM Role to an EC2 instance, the instance will have the access to other AWS resources.

In this example, the EC2 is assigned a IAM Role accessing to EMR.

Try in system

EC2

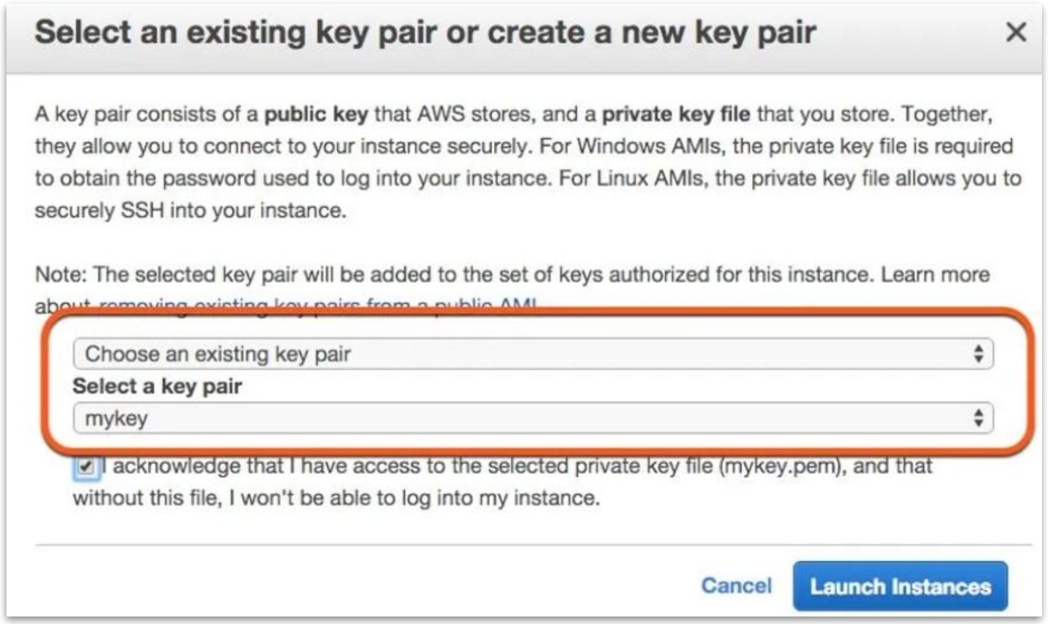
AWS Compute



Key pair name	Fingerprint
mykey	cd:5a:e6:97:73:2b:09:48:3a:60:7d:ba:80:7d:d4:e6:aa:26:68:93

Key Pair: mykey

Key pair name mykey
Fingerprint cd:5a:e6:97:73:2b:09:48:3a:60:7d:ba:80:7d:d4:e6:aa:26:68:93



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Choose an existing key pair

Select a key pair

mykey

☑ I acknowledge that I have access to the selected private key file (mykey.pem), and that without this file, I won't be able to log into my instance.

Key Pairs

A key pair, consisting of a public key and a private key, is a set of security credentials that you use to prove your identity when connecting to an Amazon EC2 instance from SSH.

```
ssh -i mykey.pem ec2-user@ec2-198-51-100-1.compute-1.amazonaws.com
```

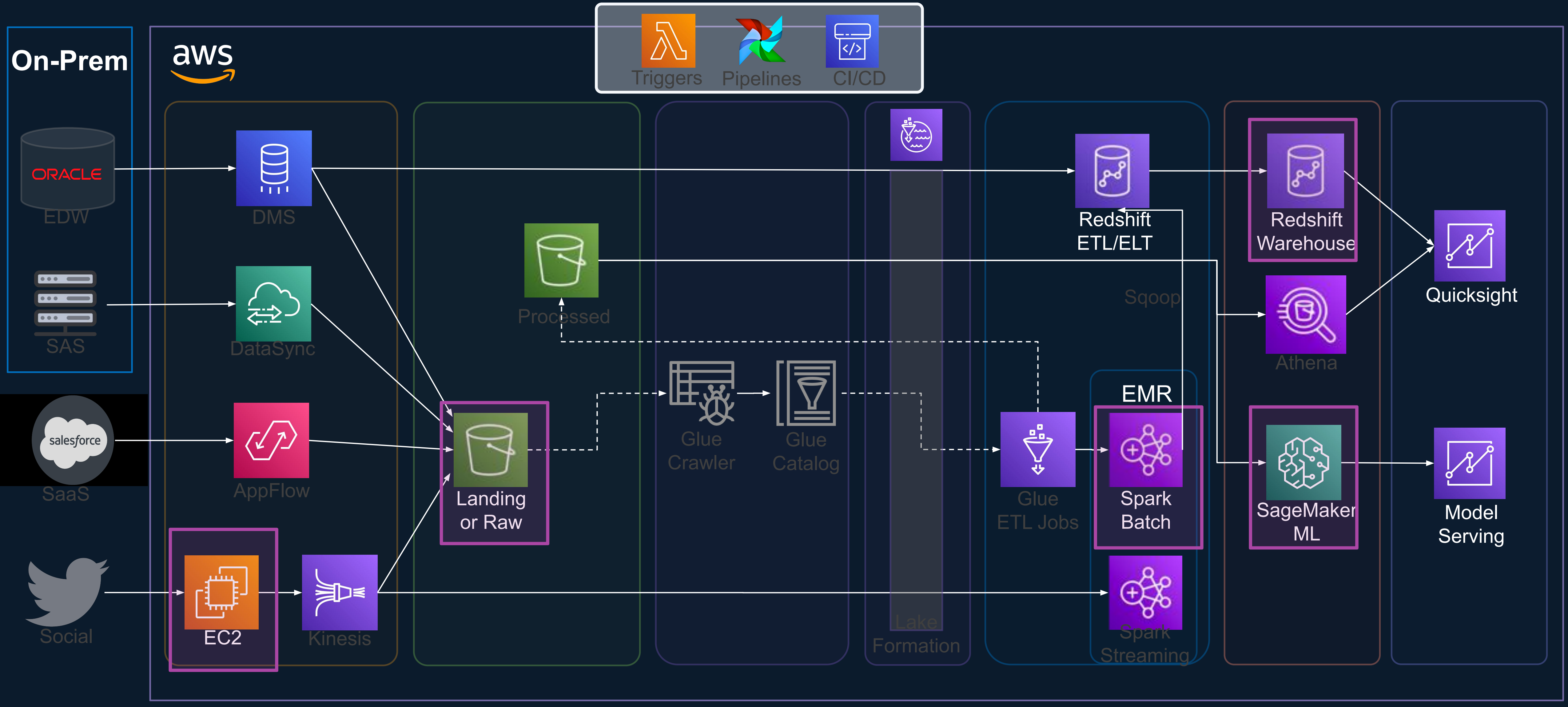
WeCloud**Data**

EC2 Lab

Demo

Demo

Launch an EC2 Instance
Connect via SSH



Data Sources | Ingestion | Data Lake | Catalog | Processing | Analytics | Apps

AWS
S3 (Simple Storage Service)

AWS Services

User and IAM Role

Compute: EC2

Storage: S3

AWS CLI and APIs

Agenda.

Demo

An existing S3 Bucket

S3 Labs

AWS Storage

Demo

Demo

Create S3 Bucket in Management Console

S3

AWS Compute

What is S3?

Amazon Simple Storage Service (Amazon S3) is an object storage service. Customers can use S3 to store and protect any amount of data for a range of use cases, such as data lakes, backup and restore, archive, enterprise applications, IoT devices, and big data analytics.

Why use S3?

- Use S3 as data lake to store all kinds of data.
- Use S3 as storage of data warehouses.
- Use S3 as files folder to store programing scripts.

S3 Pricing

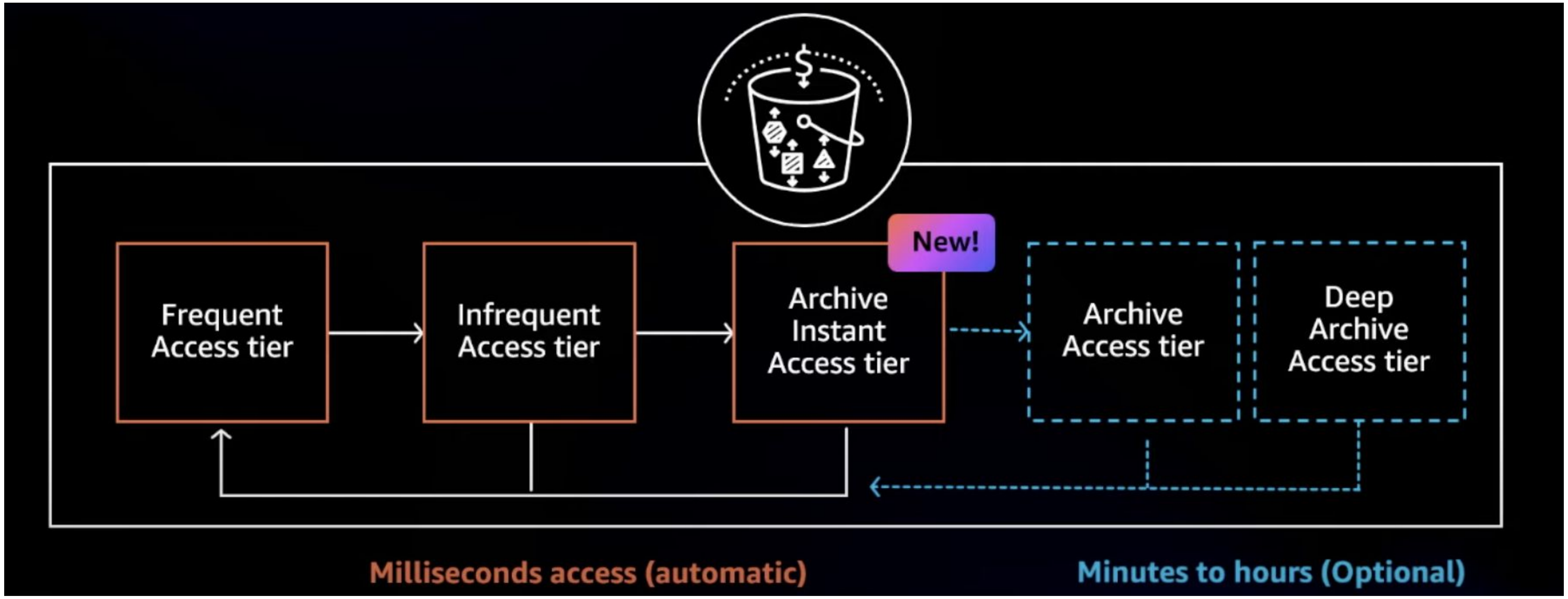
Pay only for what you use. There is no minimum charge. There are six Amazon S3 cost components to consider:

- storage
- request and data retrieval
- data transfer
- management and analytics
- replication pricing
- S3 Object Lambda

Useful link: https://aws.amazon.com/s3/pricing/

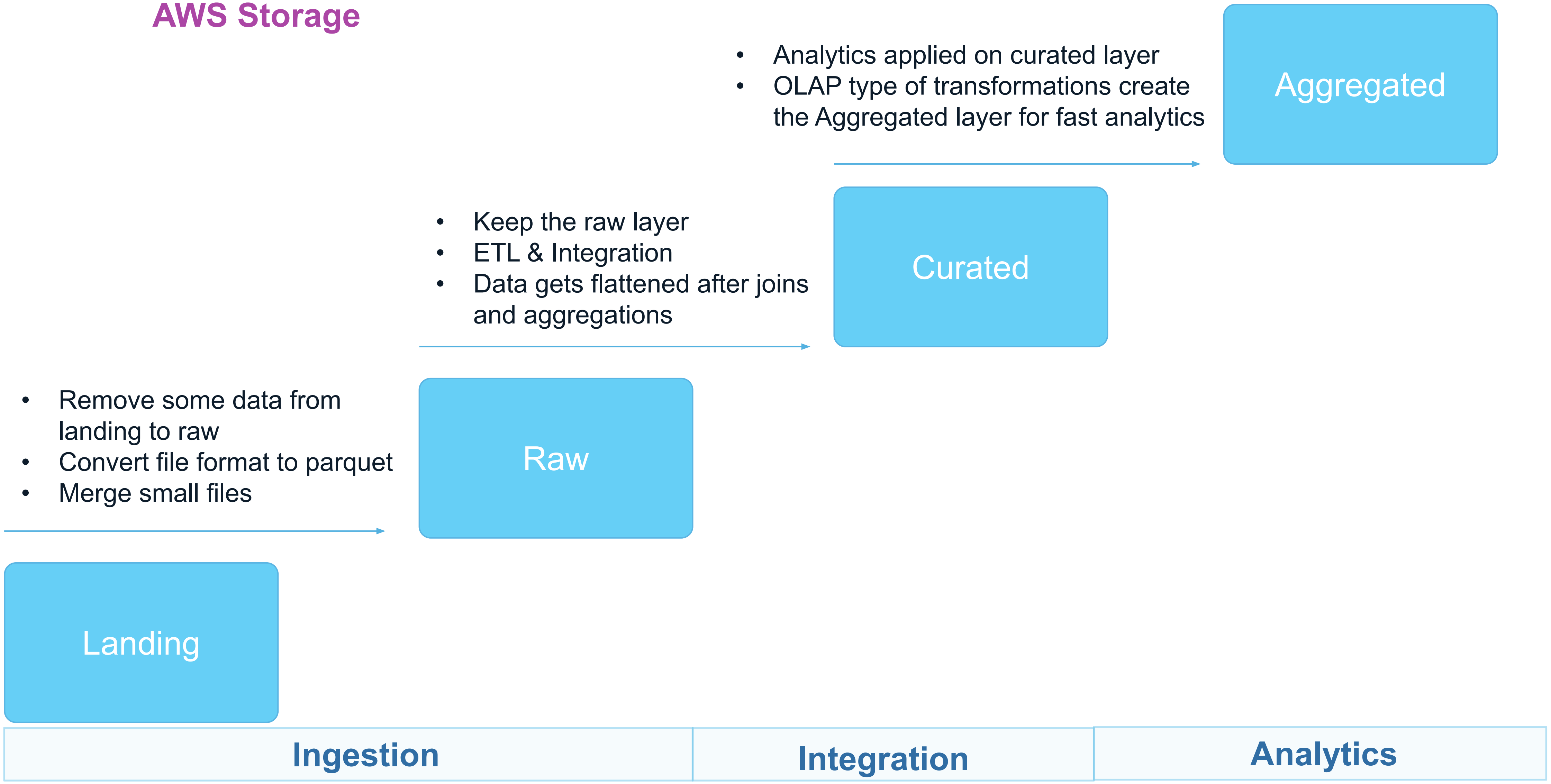
S3 Intelligent Tiers

AWS Storage



Different Layers of Data Storage

AWS Storage



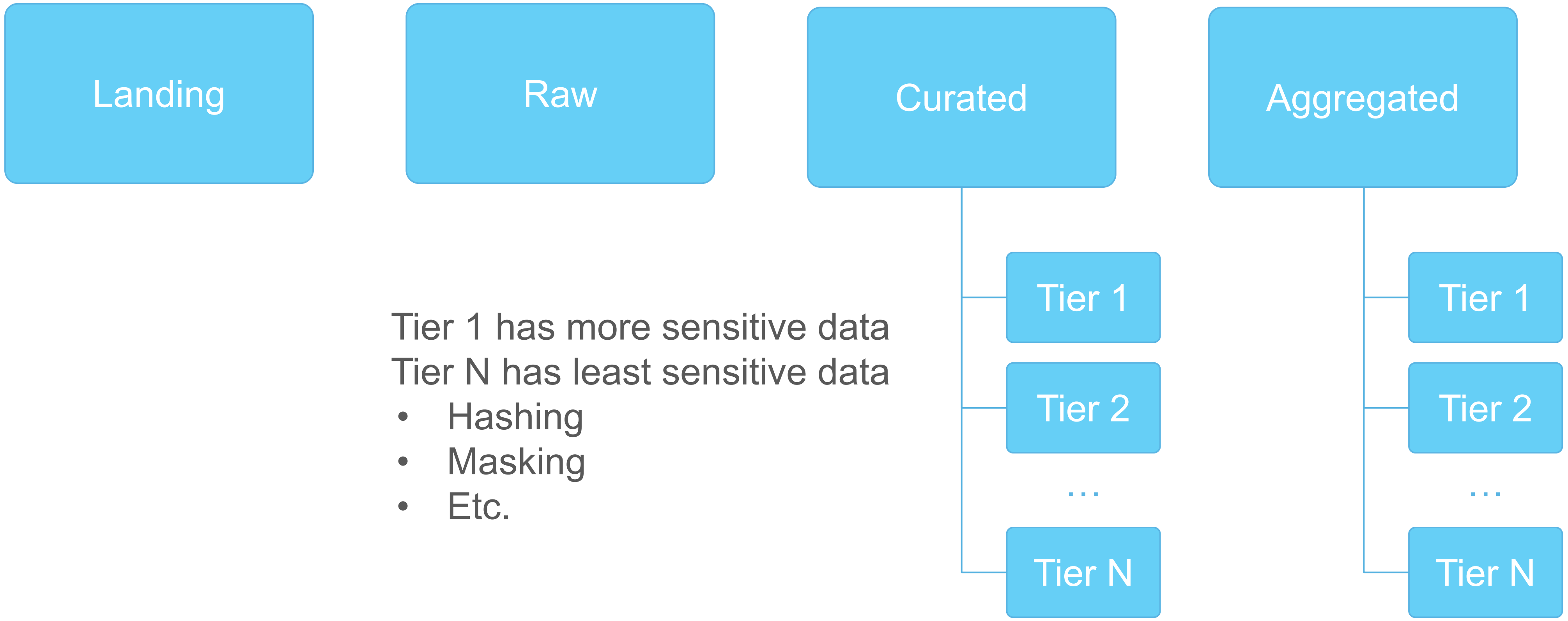
- Analytics applied on curated layer
- OLAP type of transformations create the Aggregated layer for fast analytics

- Keep the raw layer
- ETL & Integration
- Data gets flattened after joins and aggregations

- Remove some data from landing to raw
- Convert file format to parquet
- Merge small files

Different Layers of Data Sensitivity

AWS Storage



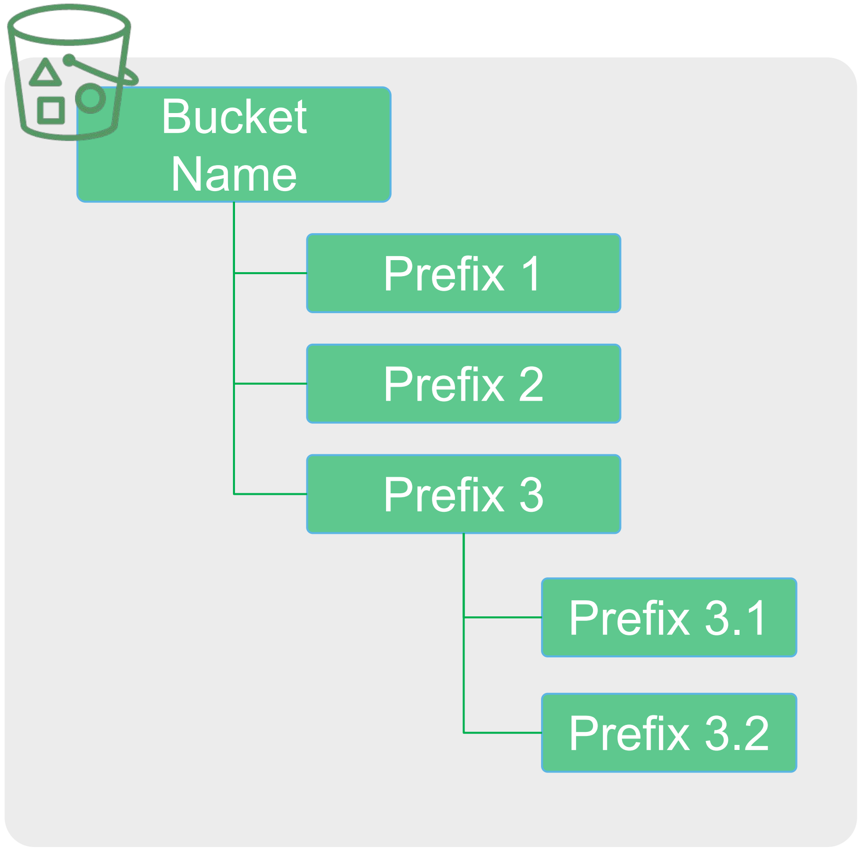
Tier 1 has more sensitive data
Tier N has least sensitive data

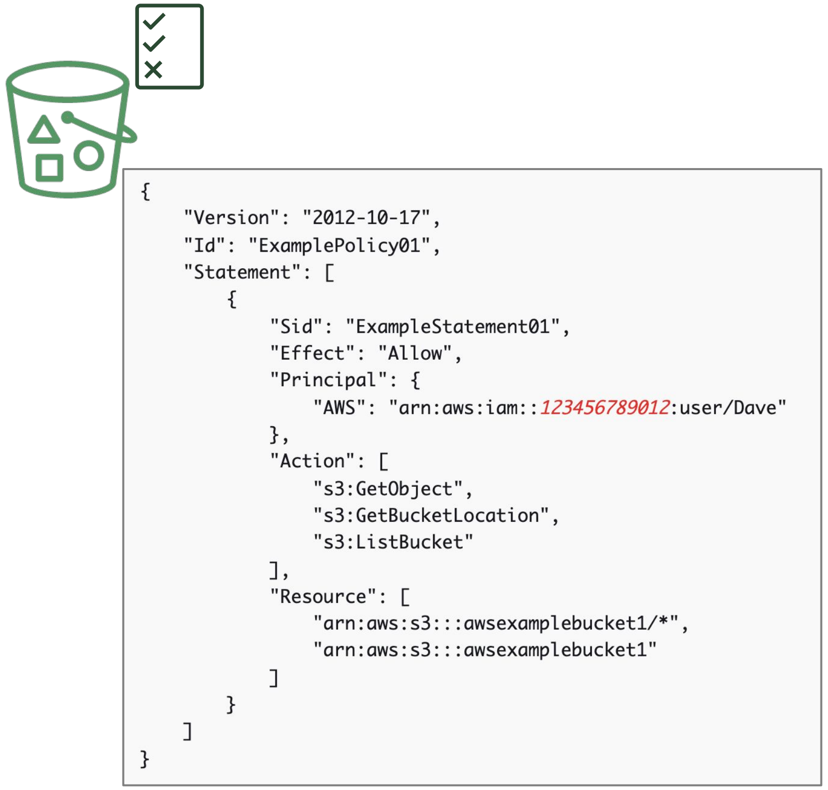
- Hashing
- Masking
- Etc.

https://youtu.be/XpTly4XHmqc

Data Lake Bucket Strategy

AWS Storage





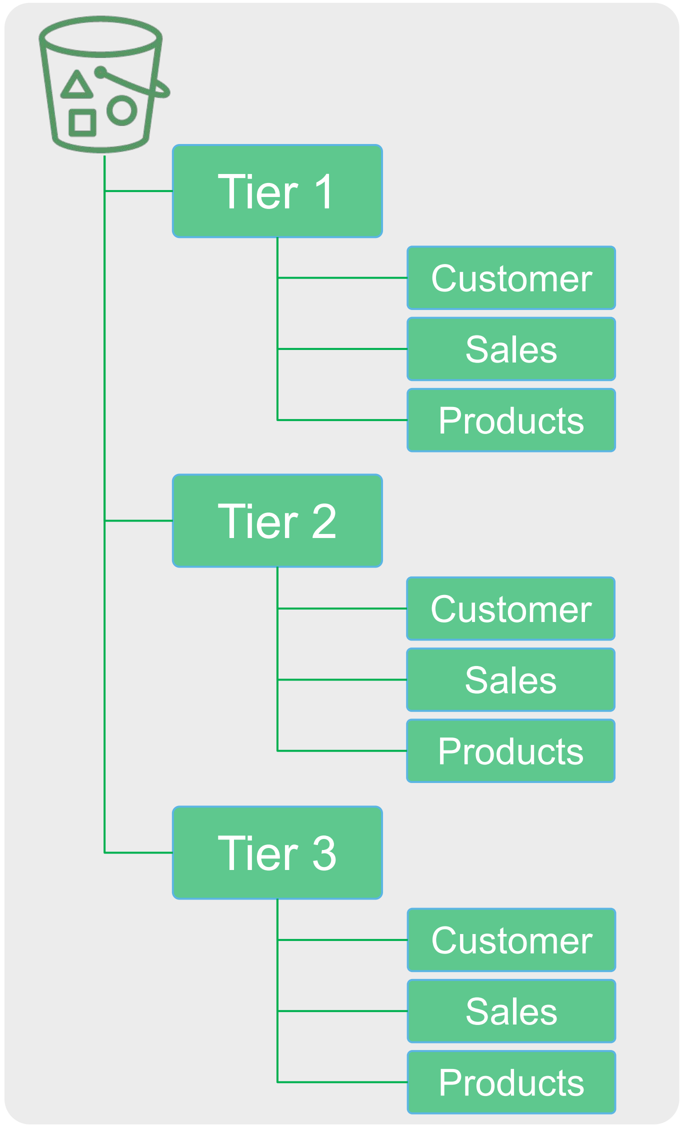
```
{
    "Version": "2012-10-17",
    "Id": "ExamplePolicy01",
    "Statement": [
        {
            "Sid": "ExampleStatement01",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": [
                "s3:GetObject",
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::awsexamplebucket1/*",
                "arn:aws:s3:::awsexamplebucket1"
            ]
        }
    ]
}
```

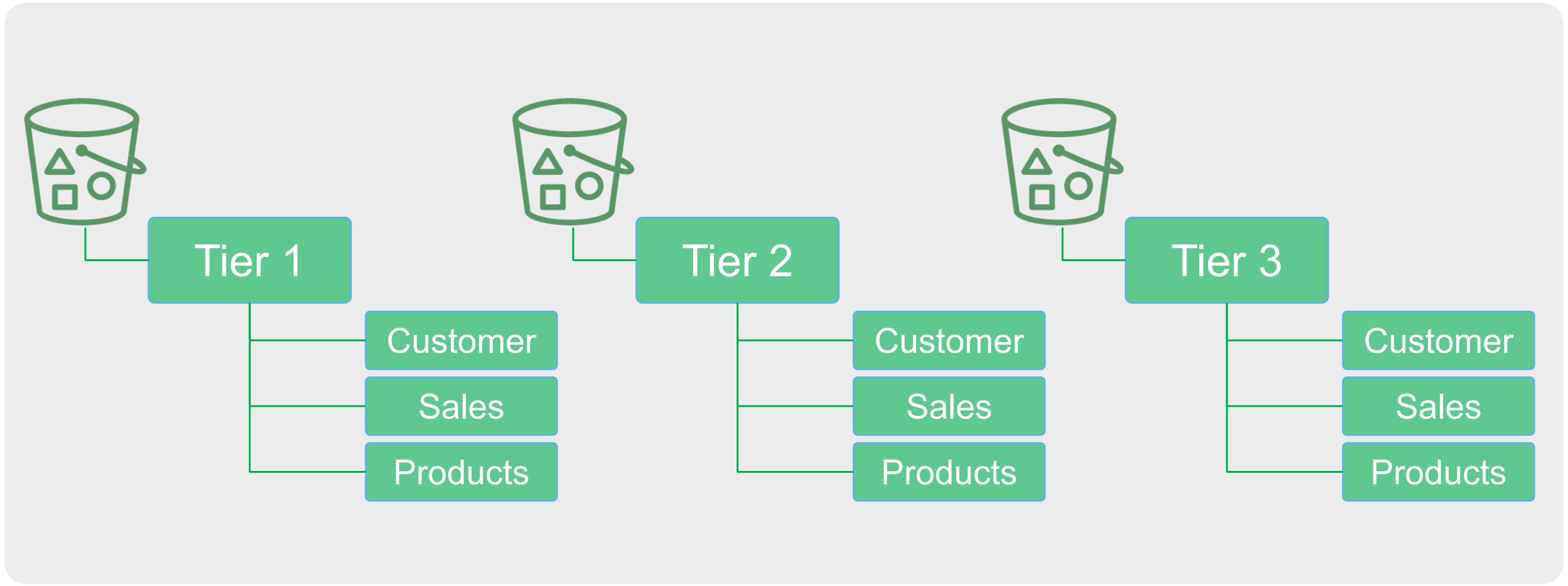
S3 bucket policy can is boring and can get messy

https://youtu.be/XpTly4XHmqc

Data Lake Bucket Strategy

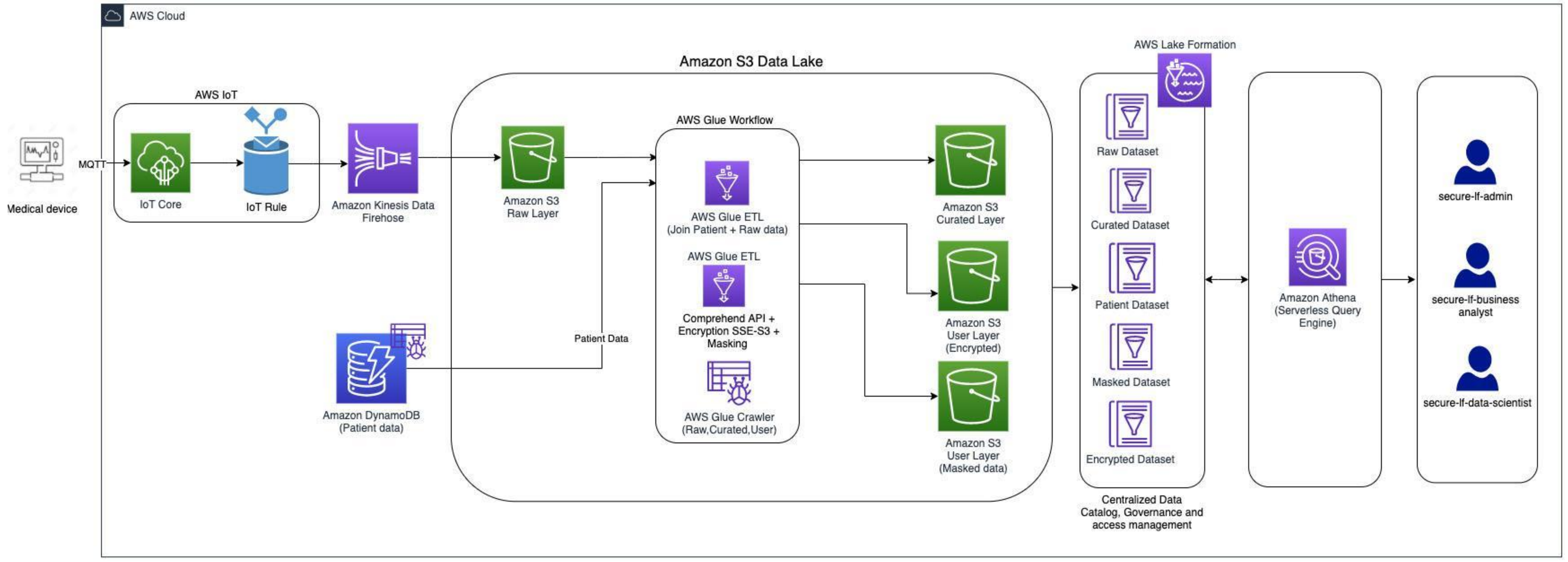
AWS Storage





Landing Zones

AWS Storage



AWS Cloud
Medical device
MQTT
AWS IoT
IoT Core
IoT Rule
Amazon Kinesis Data Firehose
Amazon S3 Data Lake
Amazon S3 Raw Layer
AWS Glue Workflow
AWS Glue ETL (Join Patient + Raw data)
AWS Glue ETL
Comprehend API + Encryption SSE-S3 + Masking
AWS Glue Crawler (Raw,Curated,User)
Patient Data
Amazon DynamoDB (Patient data)
Amazon S3 Curated Layer
Amazon S3 User Layer (Encrypted)
Amazon S3 User Layer (Masked data)
AWS Lake Formation
Raw Dataset
Curated Dataset
Patient Dataset
Masked Dataset
Encrypted Dataset
Centralized Data Catalog, Governance and access management
Amazon Athena (Serverless Query Engine)
secure-lf-admin
secure-lf-business analyst
secure-lf-data-scientist

https://aws.amazon.com/blogs/big-data/create-a-secure-data-lake-by-masking-encrypting-data-and-enabling-fine-grained-access-with-aws-lake-formation/

WeCloudData

Discussion: S3 Use Cases

AWS Storage

- Data Lake
- Code
- Audit files
- Log files
- ML model files
- Templates
- Scripts (pipelines)

AWS
CLI & APIs

AWS Services

User and IAM Role

Compute: EC2

Storage: S3

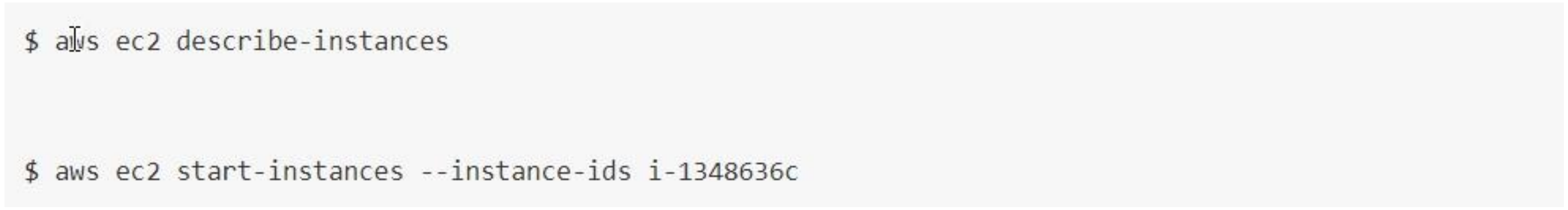
CLI and APIs

Agenda.

CLI and APIs

What is AWS CLI?

The AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.

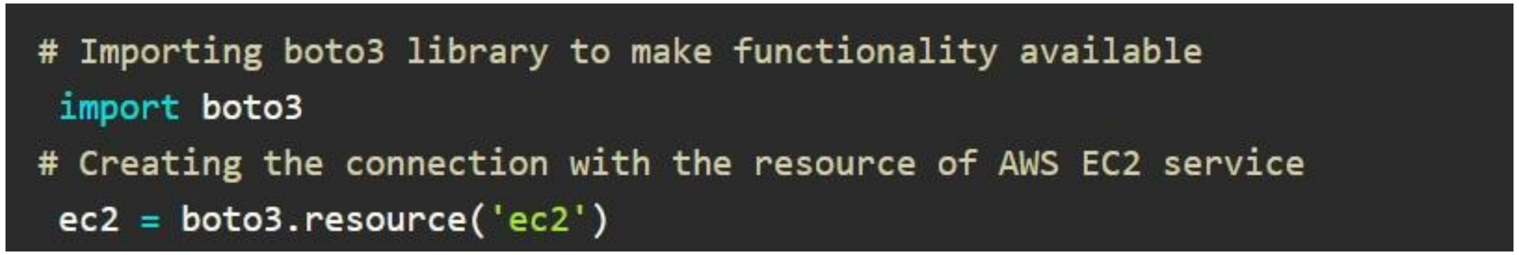


```
$ aws ec2 describe-instances


$ aws ec2 start-instances --instance-ids i-1348636c
```

What is AWS Python API ?

The AWS SDK for Python (Boto3) provides a Python API for AWS infrastructure services. Using the SDK for Python, you can build applications on top of Amazon S3, Amazon EC2, Amazon DynamoDB, and more.



```
# Importing boto3 library to make functionality available
 import boto3
# Creating the connection with the resource of AWS EC2 service
 ec2 = boto3.resource('ec2')
```

CLI Syntax

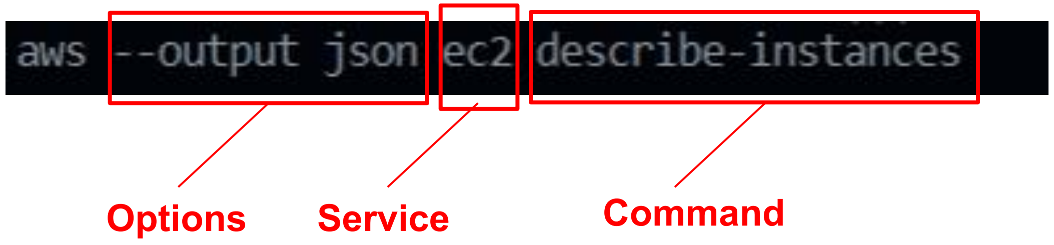
Available Services

AWS CLI is almost available for any services in AWS. Please refer to this link to check CLI details of each service. https://docs.aws.amazon.com/cli/latest/reference/.

The result of a CLI command can be in 3 format option: json, text, table. We usually use **json**.

CLI Command Structure

```
aws [options] <service> <subcommand> [parameters]
```



```
aws --output json ec2 describe-instances
```

Try in system

Set CLI in Local System

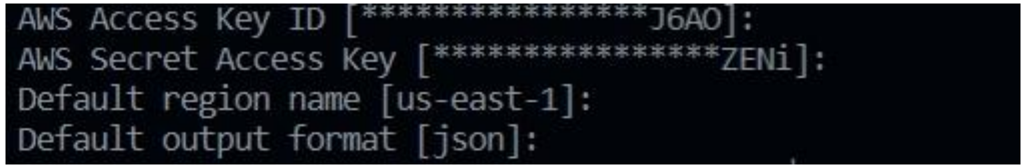
CLI Installation

- https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

CLI configuration

- You need to set your aws secret credentials and Regions in you local system.

```
aws configure
```



```
AWS Access Key ID [****************J6AO]:
AWS Secret Access Key [****************ZENi]:
Default region name [us-east-1]:
Default output format [json]:
```

Try in system

CLI Command Examples – S3

AWS CLI S3 Commands

- https://docs.aws.amazon.com/cli/latest/reference/s3/index.html

AWS CLI S3 Commands Example

```
# s3 make bucket (create bucket)
aws s3 mb s3://tgsbucket --region us-west-2

# s3 remove bucket
aws s3 rb s3://tgsbucket --force

# s3 list
aws s3 ls s3://tgsbucket

# s3 copy
aws s3 cp s3://tgsbucket/getdata.php /local/dir/data

# s3 move
aws s3 mv s3://tgsbucket/source.json s3://backup-bucket

# s3 remove files
aws s3 rm s3://tgsbucket/queries.txt
```

Try in system

boto3 (Python API)

Available Services

boto3 allows Python developers to write software that makes use of services like Amazon S3 and Amazon EC2. boto3 documents: https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html.

boto3 installation

```
pip install boto3
```

Use boto3

```
import boto3

s3_client = boto3.client('s3')
s3_client.create_bucket(Bucket=bucket_name)
```

Try in system

boto3 – S3

Class - client

Clients provide a low-level interface to the AWS service. Their definitions are generated by a JSON service description present in the **botocore library**.

```
import boto3

#Upload a file from local to s3 with client
s3_client = boto3.client('s3')
s3_client.upload_file('test.csv', 'weclouddata-demo-bucket', 'tmp/demo_local.csv')
```

Class - resource

Resources are a higher-level abstraction compared to clients. They are generated from a JSON resource description that is present in the boto library itself.

```
import boto3
#Upload a file from local to s3 with resource
s3 = boto3.resource("s3")
bucket=s3.Bucket('weclouddata-demo-bucket')
bucket.upload_file('/tmp/demo_local.csv", 'test.csv')
```

Try in system

boto3 – S3

Clients vs Resources

To summarize, resources are higher-level abstractions of AWS services compared to clients. **Resources are the recommended pattern to use boto3** as you don’t have to worry about a lot of the underlying details when interacting with AWS services. As a result, code written with Resources tends to be simpler.

Available services

https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/index.html.

Try in system

Demo

Working with Python SDK (Boto3):
client resource to upload a file

(Step from searching services in AWS web page)

Thank you

WeCloudData

500-80 Bloor Street West, Toronto
ON
www.weclouddata.com