



BLACKBUCKS INTERNSHIP REPORT

**"Effortless File Migration: S3 to DynamoDB
Automation with AWS Lambda and IAM Role"**

SUBMITTED BY

K. TEJA VENKATA VINESH KUMAR (20B915424)
E. JAHNAVI (20B91A5413)

UNDER THE GUIDANCE OF MR. AASHU DEV

**Blackbuck Engineers Pvt. Ltd
Road No 36, Jubilee Hills, Hyderabad**

Team Members:

- Kanchala Teja Venkata Vinesh Kumar (20B91A5424)
- Erapani Jahnavi (20B91A5413)

Title:

"Effortless File Migration: S3 to DynamoDB Automation with AWS Lambda and IAM Role"

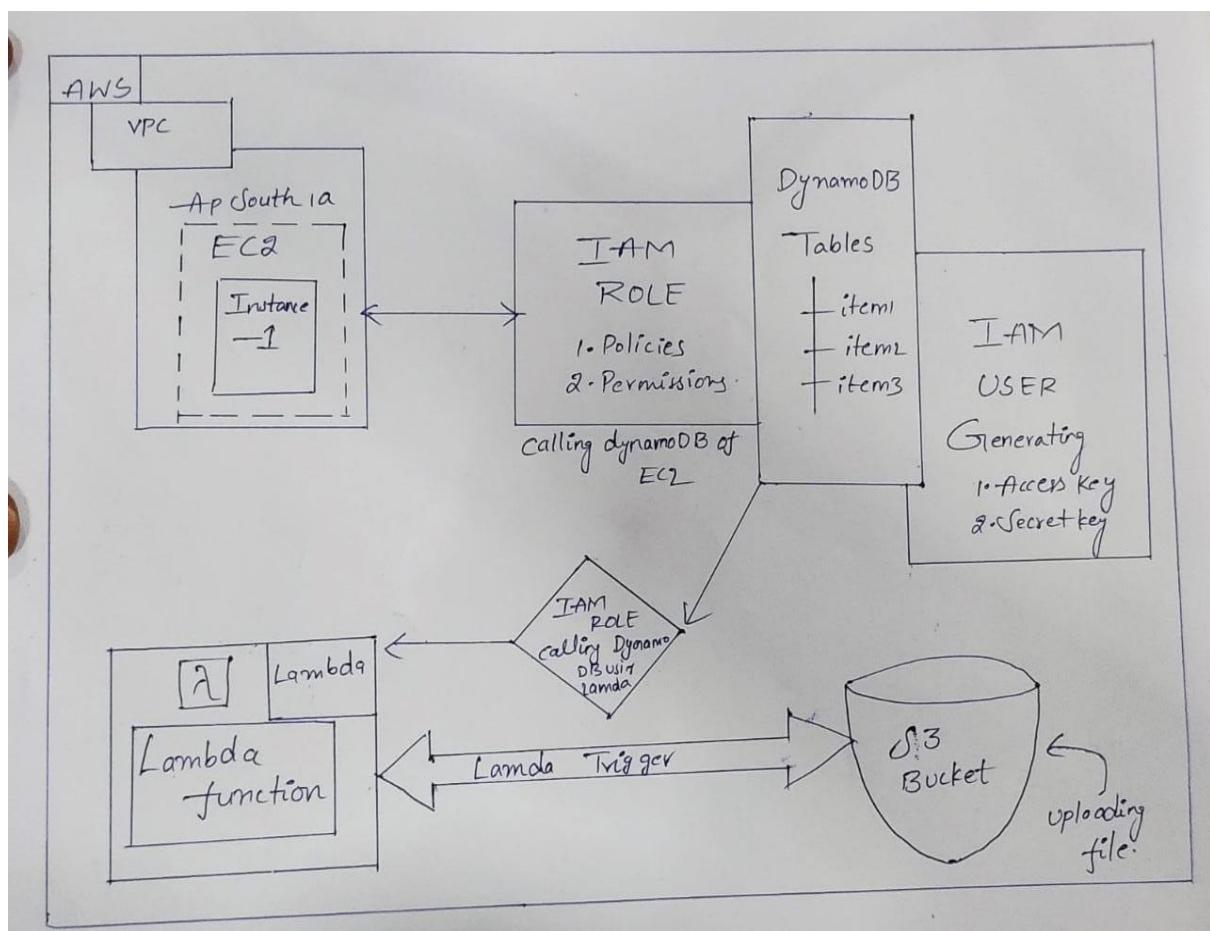
Services used:

- EC2 (Elastic Compute Cloud)
- VPC (Virtual Private Cloud)
- S3 (Simple Storage Service)
- DynamoDB
- IAM (Identity and Access Management)
- AWS Lambda

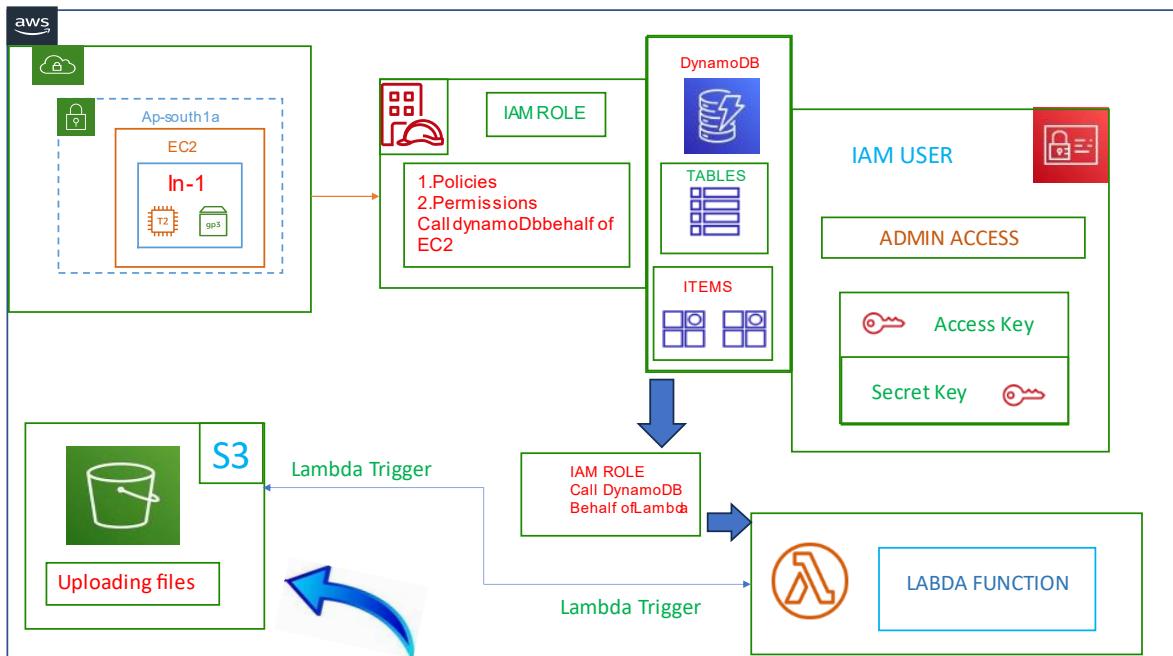
Description:

The below architecture automates the transfer of files from Amazon S3 to DynamoDB using AWS Lambda and IAM roles. It involves creating an IAM role with appropriate permissions, configuring a Lambda function triggered by S3 events, and implementing code to read files from S3, process them, and write to DynamoDB. The Lambda function is associated with the IAM role to ensure proper access. Through this automation, files can seamlessly move between S3 and DynamoDB, eliminating manual intervention. The solution enhances efficiency, reduces errors, and facilitates data integration, providing a streamlined and automated file transfer process.

- Rough architecture



- **Final architecture**



Cloud computing

Cloud computing is on-demand access, via the internet, to computing resources—applications, servers (physical servers and virtual servers), data storage, development tools, networking capabilities, and more—hosted at a remote data center managed by a cloud services provider (or CSP). The CSP makes these resources available for a monthly subscription fee or bills them according to usage.

Compared to traditional on-premises IT, and depending on the cloud services you select, cloud computing helps do the following:

- **Lower IT costs:** Cloud lets you offload some or most of the costs and effort of purchasing, installing, configuring, and managing your own on-premises infrastructure.
- **Improve agility and time-to-value:** With cloud, your organization can start using enterprise applications in minutes, instead of waiting weeks or months for IT to respond to a request, purchase and configure supporting hardware, and install software. Cloud also lets

you empower certain users—specifically developers and data scientists.

- **Scale more easily and cost-effectively:** Cloud provides elasticity—instead of purchasing excess capacity that sits unused during slow periods, you can scale capacity up and down in response to spikes and dips in traffic. You can also take advantage of your cloud provider’s global network to spread your applications closer to users around the world.

The term ‘cloud computing’ also refers to the technology that makes cloud work. This includes some form of virtualized IT infrastructure—servers, operating system software, networking, and other infrastructure that’s abstracted, using special software, so that it can be pooled and divided irrespective of physical hardware boundaries. For example, a single hardware server can be divided into multiple virtual servers.

Cloud Computing Services:

- IaaS (Infrastructure-as-a-Service)
- PaaS (Platform-as-a-Service)
- SaaS (Software-as-a-service)

are the three most common models of cloud services, and it’s not uncommon for an organization to use all three.

IaaS (Infrastructure-as-a-Service)

IaaS provides on-demand access to fundamental computing resources—physical and virtual servers, networking, and storage—over the internet on a pay-as-you-go basis. IaaS enables end users to scale and shrink resources on an as-needed basis, reducing the need for high, up-front capital expenditures or unnecessary on-premises or ‘owned’ infrastructure and for overbuying resources to accommodate periodic spikes in usage.

In contrast to SaaS and PaaS (and even newer PaaS computing models such as containers and serverless), IaaS provides the users with the lowest-level control of computing resources in the cloud.

IaaS was the most popular cloud computing model when it emerged in the early 2010s. While it remains the cloud model for many types of workloads, use of SaaS and PaaS is growing at a much faster rate.

PaaS (Platform-as-a-service)

PaaS provides software developers with on-demand platform—hardware, complete software stack, infrastructure, and even development tools—for running, developing, and managing applications without the cost, complexity, and inflexibility of maintaining that platform on-premises.

With PaaS, the cloud provider hosts everything—servers, networks, storage, operating system software, middleware, databases—at their data center. Developers simply pick from a menu to ‘spin up’ servers and environments they need to run, build, test, deploy, maintain, update, and scale applications.

Today, PaaS is often built around containers, a virtualized compute model one step removed from virtual servers. Containers virtualize the operating system, enabling developers to package the application with only the operating system services it needs to run on any platform, without modification and without need for middleware.

SaaS (Software-as-a-Service)

SaaS—also known as cloud-based software or cloud applications—is application software that’s hosted in the cloud, and that user’s access via a web browser, a dedicated desktop client, or an API that integrates with a desktop or mobile operating system. In most cases, SaaS users pay a monthly or annual subscription fee; some may offer ‘pay-as-you-go’ pricing based on your actual usage.

In addition to the cost savings, time-to-value, and scalability benefits of cloud, SaaS offers the following:

- **Automatic upgrades:** With SaaS, users take advantage of new features as soon as the provider adds them, without having to orchestrate an on-premises upgrade.
- **Protection from data loss:** Because SaaS stores application data in the cloud with the application, users don’t lose data if their device crashes or breaks.

SaaS is the primary delivery model for most commercial software today—there are hundreds of thousands of SaaS solutions available, from the most focused industry and departmental applications to powerful enterprise software database and AI (artificial intelligence) software.

Cloud Service Providers:

- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform
- Oracle
- IBM cloud
- Salesforce

Amazon Web Services:

Amazon Web Services, Inc. (AWS) is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered, pay-as-you-go basis. Oftentimes, clients will use this in combination with autoscaling (a process that allows a client to use more computing in times of high application usage, and then scale down to reduce costs when there is less traffic). These cloud computing web services provide various services related to networking, computing, storage, middleware, IoT and other processing capacity, as well as software tools via AWS server farms. This frees clients from managing, scaling, and patching hardware, and operating systems.

One of the foundational services is Amazon Elastic Compute Cloud (EC2), which allows users to have at their disposal a virtual cluster of computers, with extremely high availability, which can be interacted with over the internet via REST APIs, a CLI or the AWS console. AWS's virtual computers emulate most of the attributes of a real computer, including hardware central processing units (CPUs) and graphics processing units (GPUs) for processing; local/RAM memory; hard disk /SSD storage; a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, and customer relationship management (CRM).

AWS services are delivered to customers via a network of AWS server farms located throughout the world. Fees are based on a combination of usage (known as a "Pay-as-you-go" model), hardware, operating system, software, or networking features chosen by the subscriber required availability, redundancy, security, and service options. Subscribers can pay for a single virtual AWS computer, a dedicated physical computer, or clusters of either.

Amazon provides select portions of security for subscribers (e.g., physical security of the data centers) while other aspects of security are the responsibility of the subscriber (e.g., account management, vulnerability scanning, patching). AWS operates for many global geographical regions including seven in North America.

Amazon markets AWS to subscribers as a way of obtaining large-scale computing capacity more quickly and cheaply than building an actual physical server farm. All services are billed based on usage, but each service measures usage in varying ways. As of 2021 Q4, AWS has 33% market share for cloud infrastructure while the next two competitors Microsoft Azure and Google Cloud have 21%, and 10% respectively, according to Synergy Group.

Why AWS?

- **Easy to use:**

AWS is designed to allow application providers, ISVs, and vendors to host your applications quickly and securely – whether an existing application or a new SaaS-based application. You can use the AWS Management Console or well-documented web services APIs to access AWS's application hosting platform.

- **Flexible:**

AWS enables you to select the operating system, programming language, web application platform, database, and other services you need. With AWS, you receive a virtual environment that lets you load the software and services your application requires. This eases the migration process for existing applications while preserving options for building new solutions.

- **Cost-effective:**

You pay only for the compute power, storage, and other resources you use, with no long-term contracts or up-front commitments. For more information on comparing the costs of other hosting alternatives with AWS, see the AWS Economics Center.

- **Reliable:**

With AWS, you take advantage of a scalable, reliable, and secure global computing infrastructure, the virtual backbone of Amazon.com's multi-billion-dollar online business that has been honed for over a decade.

- **Scalable and High performance:**

Using AWS tools, Auto Scaling, and Elastic Load Balancing, your application can scale up or down based on demand. Backed by Amazon's massive infrastructure, you have access to compute and storage resources when you need them.

- **Secure:**

Using AWS tools, Auto Scaling, and Elastic Load Balancing, your application can scale up or down based on demand. Backed by Amazon's massive infrastructure, you have access to compute and storage resources when you need them.

List of AWS Services:

Amazon, the preeminent cloud vendor, broke new ground by establishing the first cloud computing service, Amazon EC2, in 2008. AWS offers more solutions and features than any other provider and has free tiers with access to the AWS Console, where users can centrally control their ministrations.

Designed around ease-of-use for various skill sets, AWS is tailored for those unaccustomed to software development utilities. Web applications can be deployed in minutes with AWS facilities, without provisioning servers or writing additional code.

- Amazon EC2 (Elastic Compute Cloud)
- Amazon RDS (Relational Database Services)
- Amazon S3 (Simple Storage Service)
- Amazon Lambda
- Amazon Cognito
- Amazon Glacier
- Amazon SNS (Simple Notification Service)
- Amazon VPC (Virtual Private Cloud)
- Amazon Lightsail
- Amazon CloudWatch
- Amazon Cloud9
- Amazon Elastic Beanstalk
- Amazon CodeCommit
- Amazon IAM (Identity and Access Management)
- Amazon Inspector
- Amazon Kinesis
- Amazon Dynamo DB
- Amazon Codecatalyst
- Amazon Kinesis
- AWS Athena
- AWS Amplify
- AWS Quicksight
- AWS Cloudformation

Amazon EC2:

Amazon Elastic Compute Cloud (EC2) is a part of Amazon.com's cloud-computing platform, Amazon Web Services (AWS), that allows users to rent virtual computers on which to run their own computer applications. EC2 encourages scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image (AMI) to configure a virtual machine, which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate server-instances as needed, paying by the second for active servers – hence the term "elastic". EC2 provides users with control over the geographical location of instances that allows for latency optimization and high levels of redundancy. In November 2010, Amazon switched its own retail website platform to EC2 and AWS.

Amazon announced a limited public beta test of EC2 on August 25, 2006, offering access on a first-come, first-served basis. Amazon added two new instance types (Large and Extra-Large) on October 16, 2007. On May 29, 2008, two more types were added, High-CPU Medium and High-CPU Extra Large. There were twelve types of instances available.

Amazon added three new features on March 27, 2008, static IP addresses, availability zones, and user selectable kernels. On August 20, 2008, Amazon added Elastic Block Store (EBS) This provides persistent storage, a feature that had been lacking since the service was introduced.

Instance types:

Initially, EC2 used Xen virtualization exclusively. However, on November 6, 2017, Amazon announced the new C5 family of instances that were based on a custom architecture around the KVM hypervisor, called Nitro. Each virtual machine, called an "instance", functions as a virtual private server. Amazon sizes instances based on "Elastic Compute Units". The performance of otherwise identical virtual machines may vary. On November 28, 2017, AWS announced a bare-metal instance type offering marking a remarkable departure from exclusively offering virtualized instance types.

As of January 2019, the following instance types were offered:

- General Purpose: A1, T3, T2, M5, M5a, M4, T3a
- Compute Optimized: C5, C5n, C4
- Memory Optimized: R5, R5a, R4, X1e, X1, High Memory, z1d
- Accelerated Computing: P3, P2, G3, F1
- Storage Optimized: H1, I3, D2

As of April 2018, the following payment methods by instance were offered:

- On-demand: pay by the hour without commitment.
- Reserved: rent instances with one-time payment receiving discounts on the hourly charge.

- Spot: bid-based service runs the jobs only if the spot price is below the bid specified by bidder. The spot price is claimed to be supply-demand based, however a 2011 study concluded that the price was generally not set to clear the market but was dominated by an undisclosed reserve price.

Amazon RDS:

Amazon Relational Database Service (or **Amazon RDS**) is a distributed relational database service by Amazon Web Services (AWS). It is a web service running "in the cloud" designed to simplify the setup, operation, and scaling of a relational database for use in applications. Administration processes like patching the database software, backing up databases and enabling point-in-time recovery are managed automatically. Scaling storage and compute resources can be performed by a single API call to the AWS control plane on-demand. AWS does not offer an SSH connection to the underlying virtual machine as part of the managed service.

Multiple Availability Zone (AZ) Deployment

In May 2010 Amazon announced Multi-Availability Zone deployment support. Amazon RDS Multi-Availability Zone (AZ) allows users to automatically provision and maintain a synchronous physical or logical "standby" replica, depending on database engine, in a different Availability Zone (independent infrastructure in a physically separate location). Multi-AZ database instance can be developed at creation time or modified to run as a multi-AZ deployment later. Multi-AZ deployments aim to provide enhanced availability and data durability for MySQL, MariaDB, Oracle, PostgreSQL and SQL Server instances and are targeted for production environments. In the event of planned database maintenance or unplanned service disruption, Amazon RDS automatically fails over to the up-to-date standby, allowing database operations to resume without administrative intervention.

Multi-AZ RDS instances are optional and have a cost associated with them. When creating a RDS instance, the user is asked if they would like to use a multi-AZ RDS instance. In Multi-AZ RDS deployments backups are done in the standby instance so I/O activity is not suspended any time, but users may experience elevated latencies for a few minutes during backups.

Read replicas.

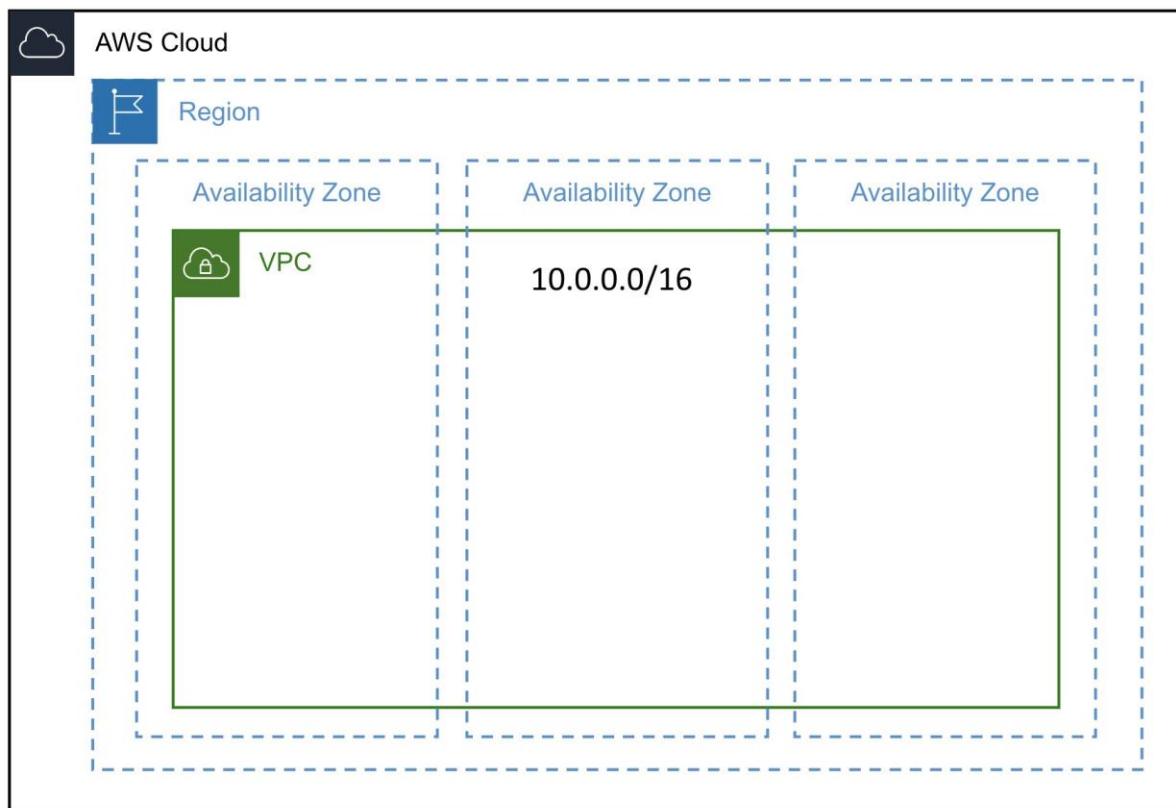
Read replicas allow different use cases such as scale in for read-heavy database workloads. There are up to five replicas available for MySQL, MariaDB, and PostgreSQL. Instances use the native, asynchronous replication functionality of their respective database engines. They have no backups configured by default and are accessible and can be used for read scaling. MySQL and MariaDB read replicas and can be made writeable again since October 2012; PostgreSQL read replicas do not support it. Replicas are done at database instance level and do not support replication at database or table level.

Performance metrics and monitoring

Performance metrics for Amazon RDS are available from the AWS Management Console or the Amazon CloudWatch API. In December 2015, Amazon announced an optional enhanced monitoring feature that provides an expanded set of metrics for the MySQL, MariaDB, and Aurora database engines.

Amazon VPC:

Amazon Virtual Private Cloud (VPC) is a commercial cloud computing service that provides a virtual private cloud, by provisioning a logically isolated section of Amazon Web Services (AWS) Cloud. Enterprise customers are able to access the Amazon Elastic Compute Cloud (EC2) over an IPsec based virtual private network. Unlike traditional EC2 instances which are allocated internal and external IP numbers by Amazon, the customer can assign IP numbers of their choosing from one or more subnets.



Amazon Web Services launched Amazon Virtual Private Cloud on 26 August 2009, which allows the Amazon Elastic Compute Cloud service to be connected to legacy infrastructure over an IPsec VPN. In AWS, the basic VPC is free to use, with users being charged by usage for additional features. EC2 and RDS instances running in a VPC can also be purchased using

Reserved Instances, however will have a limitation on resources being guaranteed.[citation needed]

IBM Cloud launched IBM Cloud VPC on 4 June 2019, provides an ability to manage virtual machine-based compute, storage, and networking resources. Pricing for IBM Cloud Virtual Private Cloud is applied separately for internet data transfer, virtual server instances, and block storage used within IBM Cloud VPC.

Google Cloud Platform resources can be provisioned, connected, and isolated in a virtual private cloud (VPC) across all GCP regions. With GCP, VPCs are global resources and subnets within that VPC are regional resources. This allows users to connect zones and regions without the use of additional networking complexity as all data travels, encrypted in transit and at rest, on Google's own global, private network. Identity management policies and security rules allow for private access to Google's storage, big data, and analytics managed services. VPCs on Google Cloud Platform leverage the security of Google's data centers.

Amazon S3:

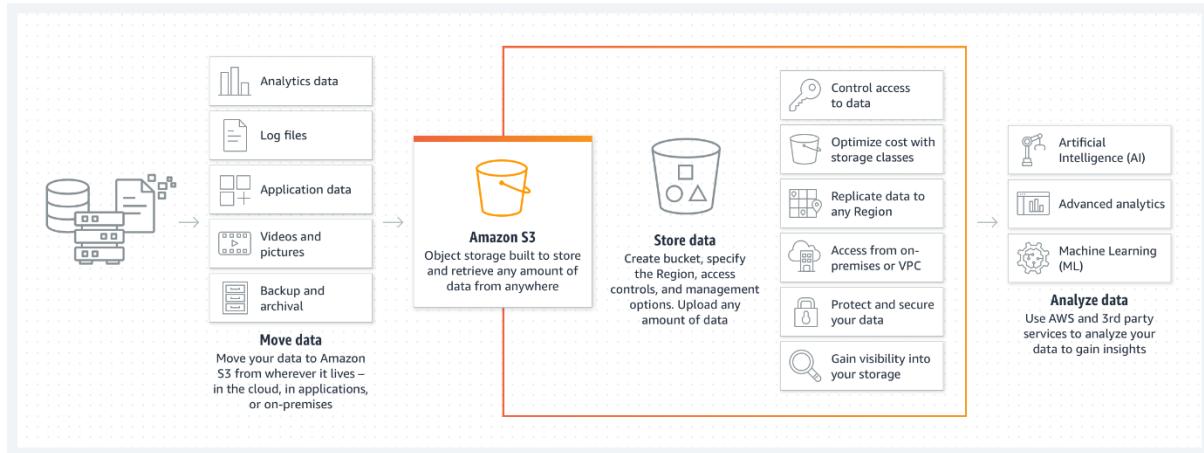
Amazon S3 manages data with an object storage architecture which aims to provide scalability, high availability, and low latency with high durability. The basic storage units of Amazon S3 are objects which are organized into buckets. Each object is identified by a unique, user-assigned key. Buckets can be managed using the console provided by Amazon S3, programmatically with the AWS SDK, or the REST application programming interface.

Objects can be up to five terabytes in size. Requests are authorized using an access control list associated with each object bucket and support versioning which is disabled by default. Since buckets are typically the size of an entire file system mount in other systems, this access control scheme is very coarse-grained. In other words, unique access controls cannot be associated with individual files. [citation needed] Amazon S3 can be used to replace static web-hosting infrastructure with HTTP client-accessible objects, index document support and error document support.

The Amazon AWS authentication mechanism allows the creation of authenticated URLs, valid for a specified amount of time. Every item in a bucket can also be served as a BitTorrent feed. The Amazon S3 store can act as a seed host for a torrent and any BitTorrent client can retrieve the file. This can drastically reduce the bandwidth cost for the download of popular objects. A bucket can be configured to save HTTP log information to a sibling bucket; this can be used in data mining operations.

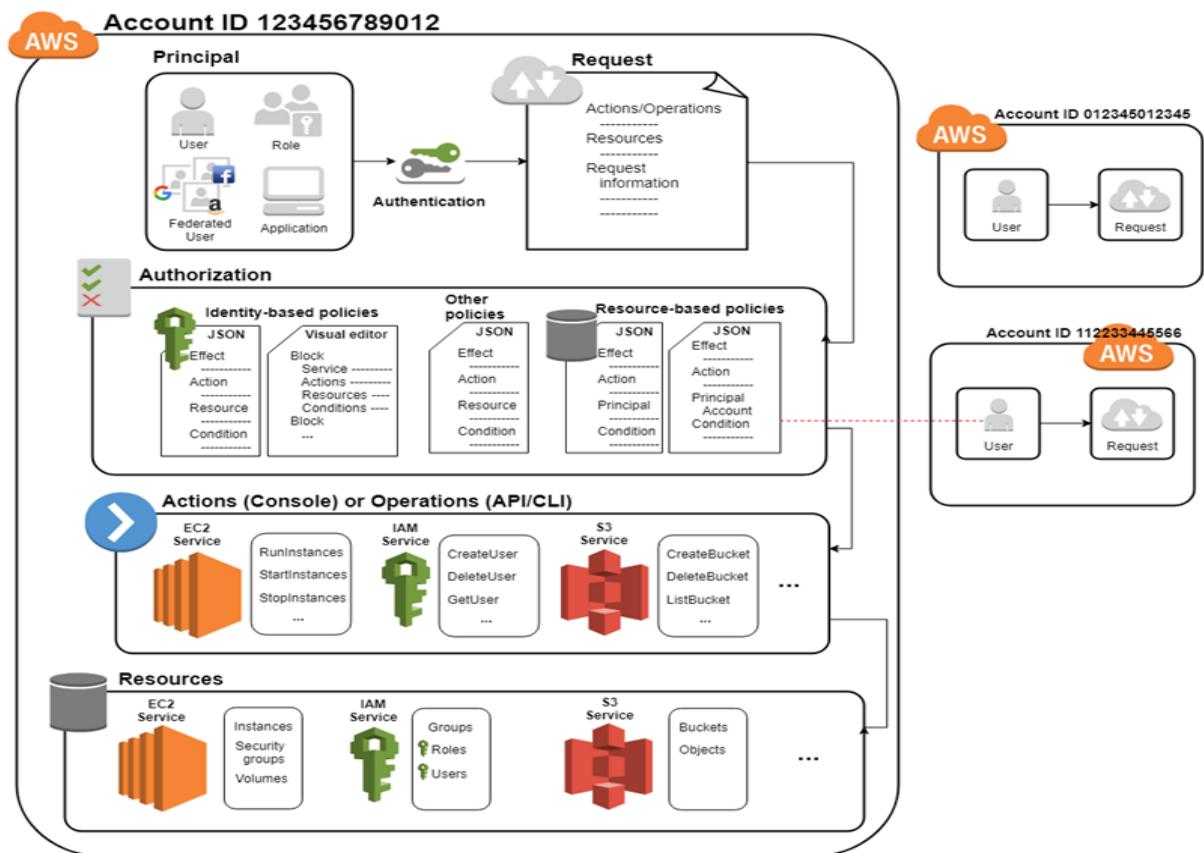
There are various User Mode File System (FUSE)-based file systems for Unix-like operating systems (for example, Linux) that can be used to mount an S3 bucket as a file system. The

semantics of the Amazon S3 file system is not that of a POSIX file system, so the file system may not behave entirely as expected.



Amazon IAM:

IAM provides the infrastructure necessary to control authentication and authorization for your AWS account. The IAM infrastructure is illustrated by the following diagram.



First, a human user or an application uses their sign-in credentials to authenticate with AWS. Authentication is provided by matching the sign-in credentials to a principal (an IAM user, federated user, IAM role, or application) trusted by the AWS account.

Next, a request is made to grant the principal access to resources. Access is granted in response to an authorization request. For example, when you first sign into the console and are on the console home page, you are not accessing a specific service. When you select a service, the request for authorization is sent to that service and it looks to see if your identity is on the list of authorized users, what policies are being enforced to control the level of access granted, and any other policies that might be in effect. Authorization requests can be made by principals within your AWS account or from another AWS account that you trust.

Once authorized, the principal can take action or perform operations on resources in your AWS account. For example, the principal could launch a new Amazon Elastic Compute Cloud instance, modify IAM group membership, or delete Amazon Simple Storage Service buckets.

The previous illustration we used specific terminology to describe how to obtain access to resources. These IAM terms are commonly used when working with AWS:

IAM Resources

The user, group, role, policy, and identity provider objects that are stored in IAM.

As with other AWS services, you can add, edit, and remove resources from IAM.

IAM Identities

The IAM resource objects that are used to identify and group. You can attach a policy to an IAM identity. These include users, groups, and roles.

IAM Entities

The IAM resource objects that AWS uses for authentication. These include IAM users and roles.

Principals

A person or application that uses the AWS account root user, an IAM user, or an IAM role to sign in and make requests to AWS. Principals include federated users and assumed roles.

Human users

Also known as human identities; the people, administrators, developers, operators, and consumers of your applications.

Workload

A collection of resources and code that delivers business value, such as an application or backend process. Can include applications, operational tools, and components.

AWS Lambda:

AWS Lambda is a compute service that lets you run code without provisioning or managing servers.

Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, and logging. With Lambda, all you need to do is supply your code in one of the language runtimes that Lambda supports.

You organize your code into Lambda functions. The Lambda service runs your function only when needed and scales automatically. You only pay for the compute time that you consume—there is no charge when your code is not running.

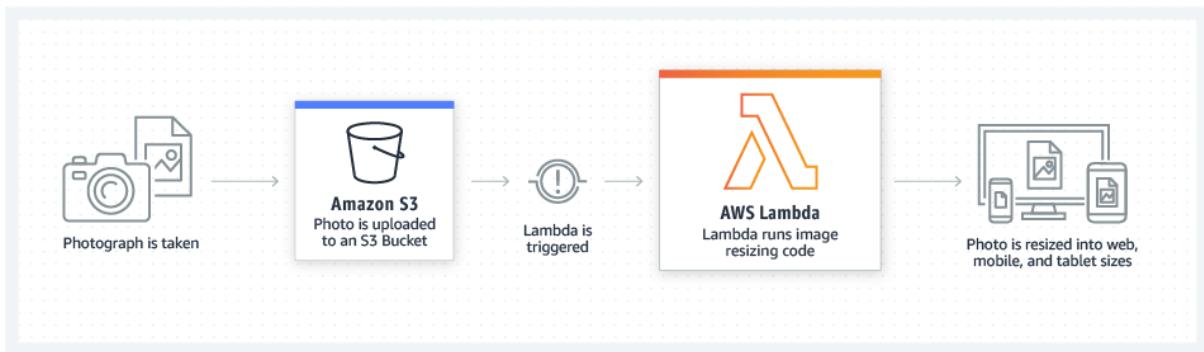
When using Lambda, you are responsible only for your code. Lambda manages the compute fleet that offers a balance of memory, CPU, network, and other resources to run your code. Because Lambda manages these resources, you cannot log in to compute instances or customize the operating system on provided runtimes.

Lambda performs operational and administrative activities on your behalf, including managing capacity, monitoring, and logging your Lambda functions.

If you do need to manage your compute resources, AWS has other compute services to consider, such as:

- AWS App Runner builds and deploys containerized web applications automatically, load balances traffic with encryption, scales to meet your traffic needs, and allows for the configuration of how services are accessed and communicate with other AWS applications in a private Amazon VPC.
- AWS Fargate with Amazon ECS runs containers without having to provision, configure, or scale clusters of virtual machines.
- Amazon EC2 lets you customize operating system, network and security settings, and the entire software stack. You are responsible for provisioning capacity, monitoring fleet health and performance, and using Availability Zones for fault tolerance.

You can use environment variables to adjust your function's behavior without updating code. An environment variable is a pair of strings that is stored in a function's version-specific configuration. The Lambda runtime makes environment variables available to your code and sets additional environment variables that contain information about the function and invocation request.

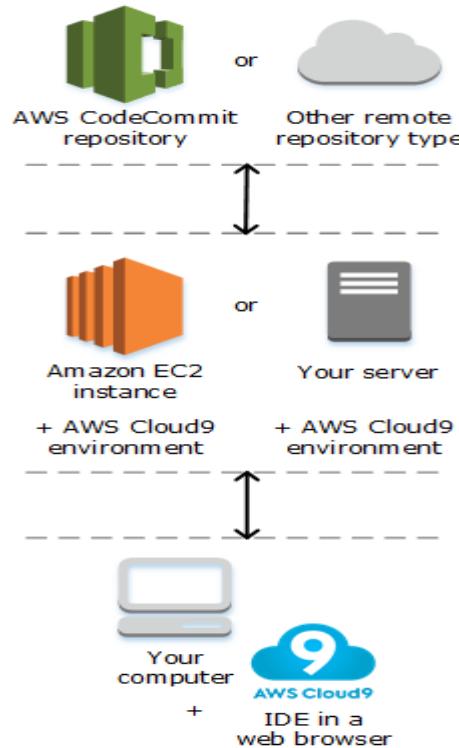


AWS Cloud9:

AWS Cloud9 is an integrated development environment, or *IDE*.

The AWS Cloud9 IDE offers a rich code-editing experience with support for several programming languages and runtime debuggers, and a built-in terminal. It contains a collection of tools that you use to code, build, run, test, and debug software, and helps you release software to the cloud.

You access the AWS Cloud9 IDE through a web browser. You can configure the IDE to your preferences. You can switch color themes, bind shortcut keys, enable programming language-specific syntax coloring and code formatting, and more.



Environments and computing resources

Behind the scenes, there are a couple of ways you can connect your environments to computing resources:

- You can instruct AWS Cloud9 to create an Amazon EC2 instance, and then connect the environment to that newly created EC2 instance. This type of setup is called an EC2 environment.
- You can instruct AWS Cloud9 to connect an environment to an existing cloud compute instance or to your own server. This type of setup is called an SSH environment.

EC2 environments and SSH environments have some similarities and some differences. If you're new to AWS Cloud9, we recommend that you use an EC2 environment because AWS Cloud9 takes care of much of the configuration for you. As you learn more about AWS Cloud9, and want to understand these similarities and differences better, see EC2 environments compared with SSH environments in AWS Cloud9.

AWS Elastic Beanstalk:

Amazon Web Services (AWS) comprises over one hundred services, each of which exposes an area of functionality. While the variety of services offers flexibility for how you want to manage your AWS infrastructure, it can be challenging to figure out which services to use and how to provision them.

With Elastic Beanstalk, you can quickly deploy and manage applications in the AWS Cloud without having to learn about the infrastructure that runs those applications. Elastic Beanstalk reduces management complexity without restricting choice or control. You simply upload your application, and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.

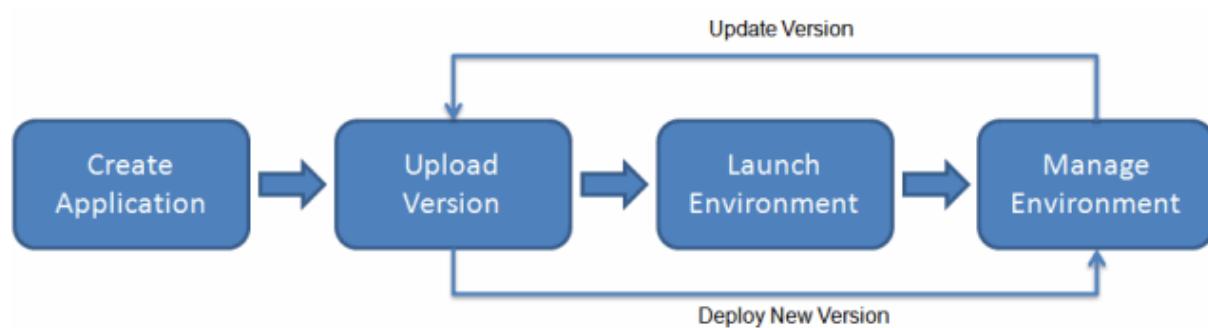
Elastic Beanstalk supports applications developed in Go, Java, .NET, Node.js, PHP, Python, and Ruby. When you deploy your application, Elastic Beanstalk builds the selected supported platform version and provisions one or more AWS resources, such as Amazon EC2 instances, to run your application.

You can interact with Elastic Beanstalk by using the Elastic Beanstalk console, the AWS Command Line Interface (AWS CLI), or **eb**, a high-level CLI designed specifically for Elastic Beanstalk.

To learn more about how to deploy a sample web application using Elastic Beanstalk, see Getting Started with AWS: Deploying a Web App.

You can also perform most deployment tasks, such as changing the size of your fleet of Amazon EC2 instances or monitoring your application, directly from the Elastic Beanstalk web interface (console).

To use Elastic Beanstalk, you create an application, upload an application version in the form of an application source bundle (for example, a Java .war file) to Elastic Beanstalk, and then provide some information about the application. Elastic Beanstalk automatically launches an environment and creates and configures the AWS resources needed to run your code. After your environment is launched, you can then manage your environment and deploy new application versions. The following diagram illustrates the workflow of Elastic Beanstalk.



AWS CodeCommit:

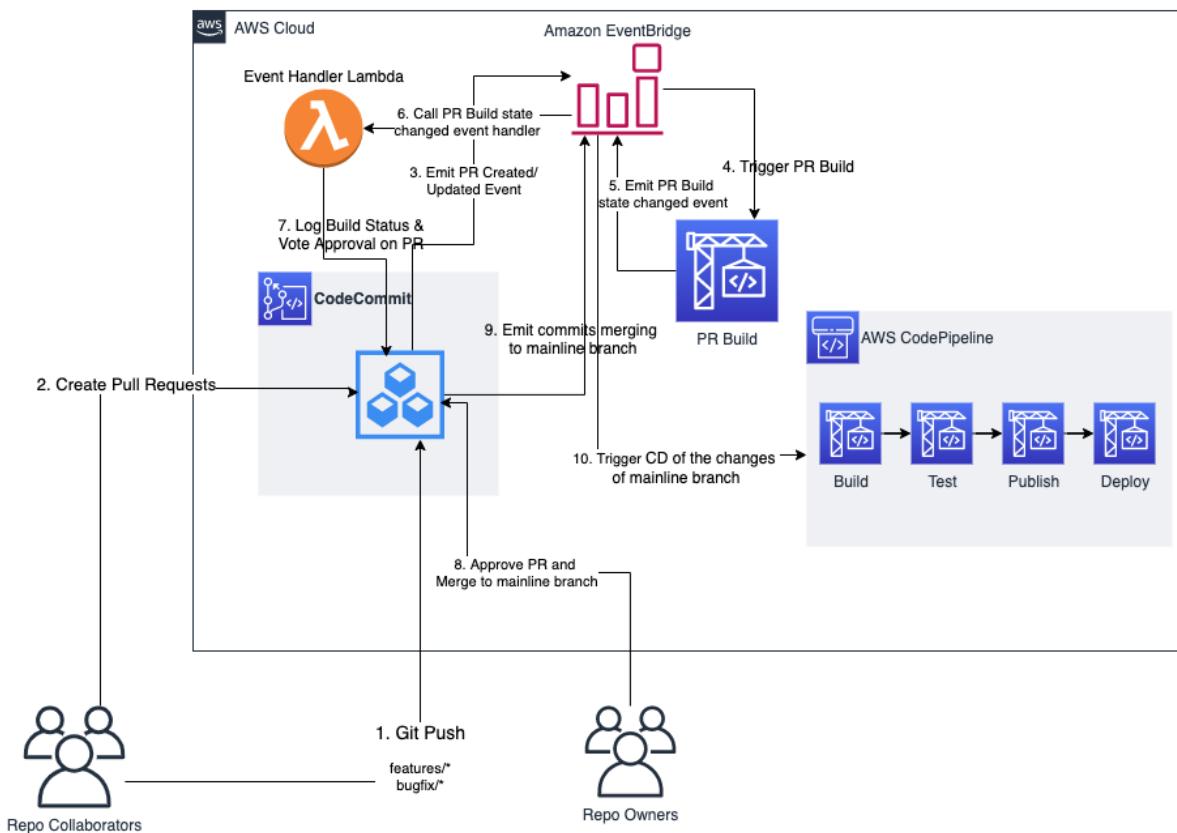
CodeCommit is a secure, highly scalable, managed source control service that hosts private Git repositories. CodeCommit eliminates the need for you to manage your own source control system or worry about scaling its infrastructure. You can use CodeCommit to store anything from code to binaries. It supports the standard functionality of Git, so it works seamlessly with your existing Git-based tools.

With CodeCommit, you can:

- **Benefit from a fully managed service hosted by AWS.** CodeCommit provides high service availability and durability and eliminates the administrative overhead of managing your own hardware and software. There is no hardware to provision and scale and no server software to install, configure, and update.
- **Store your code securely.** CodeCommit repositories are encrypted at rest as well as in transit.
- **Work collaboratively on code.** CodeCommit repositories support pull requests, where users can review and comment on each other's code changes before merging them to

branches; notifications that automatically send emails to users about pull requests and comments; and more.

- **Easily scale your version control projects.** CodeCommit repositories can scale up to meet your development needs. The service can handle repositories with large numbers of files or branches, large file sizes, and lengthy revision histories.
- **Store anything, anytime.** CodeCommit has no limit on the size of your repositories or on the file types you can store.
- **Integrate with other AWS and third-party services.** CodeCommit keeps your repositories close to your other production resources in the AWS Cloud, which helps increase the speed and frequency of your development lifecycle. It is integrated with IAM and can be used with other AWS services and in parallel with other repositories. For more information, see Product and service integrations with AWS CodeCommit.
- **Easily migrate files from other remote repositories.** You can migrate to CodeCommit from any Git-based repository.
- **Use the Git tools you already know.** CodeCommit supports Git commands as well as its own AWS CLI commands and APIs.



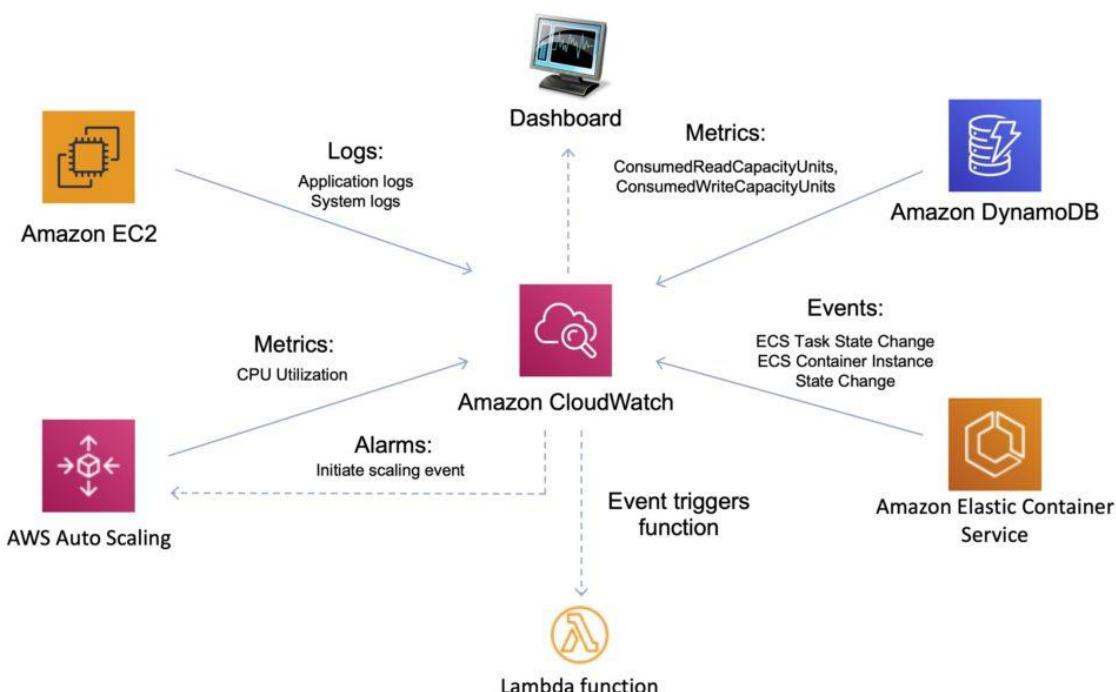
Amazon CloudWatch:

Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.

The CloudWatch home page automatically displays metrics about every AWS service you use. You can additionally create custom dashboards to display metrics about your custom applications and display custom collections of metrics that you choose.

You can create alarms that watch metrics and send notifications or automatically make changes to the resources you are monitoring when a threshold is breached. For example, you can monitor the CPU usage and disk reads and writes of your Amazon EC2 instances and then use that data to determine whether you should launch additional instances to handle increased load. You can also use this data to stop underused instances to save money.

With CloudWatch, you gain system-wide visibility into resource utilization, application performance, and operational health.



Amazon EBS (Elastic Block Store):

Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with EC2 instances. EBS volumes behave like raw, unformatted block devices. You can mount these volumes as devices on your instances. EBS volumes that are attached to an instance are exposed as storage volumes that persist independently from the life of the instance. You can create a file system on top of these volumes or use them in any way you would use a block device (such as a hard drive). You can dynamically change the configuration of a volume attached to an instance.

We recommend Amazon EBS for data that must be quickly accessible and requires long-term persistence. EBS volumes are particularly well-suited for use as the primary storage for file systems, databases, or for any applications that require fine granular updates and access to raw, unformatted, block-level storage. Amazon EBS is well suited to both database-style applications that rely on random reads and writes, and to throughput-intensive applications that perform long, continuous reads and writes.

With Amazon EBS, you pay only for what you use. For more information about Amazon EBS pricing, see the Projecting Costs Section of the Amazon Elastic Block Store page.

Features of Amazon EBS

- You create an EBS volume in a specific Availability Zone, and then attach it to an instance in that same Availability Zone. To make a volume available outside of the Availability Zone, you can create a snapshot and restore that snapshot to a new volume anywhere in that Region. You can copy snapshots to other Regions and then restore them to new volumes there, making it easier to leverage multiple AWS Regions for geographical expansion, data center migration, and disaster recovery.
- Amazon EBS provides the following volume types: General Purpose SSD, Provisioned IOPS SSD, Throughput Optimized HDD, and Cold HDD. For more information, see EBS volume types.

The following is a summary of performance and use cases for each volume type.

- General Purpose SSD volumes (gp2 and gp3) balance price and performance for a wide variety of transactional workloads. These volumes are ideal for use cases such as boot volumes, medium-size single instance databases, and development and test environments.
- Provisioned IOPS SSD volumes (io1 and io2) are designed to meet the needs of I/O-intensive workloads that are sensitive to storage performance and consistency. They provide a consistent IOPS rate that you specify when you create the volume. This enables you to predictably scale to tens of thousands of IOPS per instance. Additionally, io2 volumes provide the highest levels of volume durability.

- Throughput Optimized HDD volumes (st1) provide low-cost magnetic storage that defines performance in terms of throughput rather than IOPS. These volumes are ideal for large, sequential workloads such as Amazon EMR, ETL, data warehouses, and log processing.
- Cold HDD volumes (sc1) provide low-cost magnetic storage that defines performance in terms of throughput rather than IOPS. These volumes are ideal for large, sequential, cold-data workloads. If you require infrequent access to your data and are looking to save costs, these volumes provide inexpensive block storage.
- You can create your EBS volumes as encrypted volumes, in order to meet a wide range of data-at-rest encryption requirements for regulated/audited data and applications. When you create an encrypted EBS volume and attach it to a supported instance type, data stored at rest on the volume, disk I/O, and snapshots created from the volume are all encrypted. Encryption occurs on the servers that host EC2 instances, providing encryption of data-in-transit from EC2 instances to EBS storage. For more information, see [Amazon EBS encryption](#).
- Performance metrics, such as bandwidth, throughput, latency, and average queue length, are available through the AWS Management Console. These metrics, provided by Amazon CloudWatch, allow you to monitor the performance of your volumes to make sure that you are providing enough performance for your applications without paying for resources you don't need.

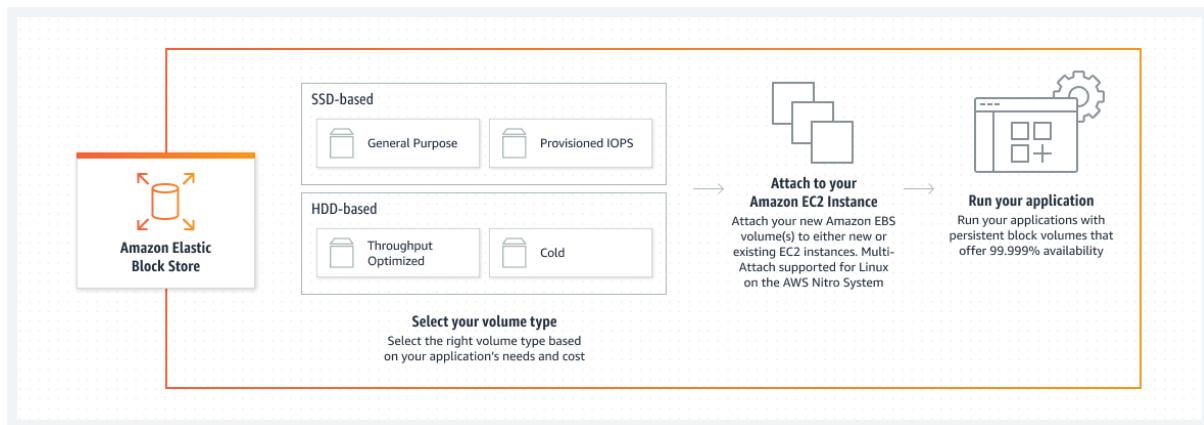


Fig. High-Performance Block Storage

Amazon Aurora:

Amazon Aurora (Aurora) is a fully managed relational database engine that's compatible with MySQL and PostgreSQL. You already know how MySQL and PostgreSQL combine the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases. The code, tools, and applications you use today with your existing MySQL and PostgreSQL databases can be used with Aurora. With some workloads, Aurora can deliver up to five times the throughput of MySQL and up to three times the throughput of PostgreSQL without requiring changes to most of your existing applications.

Aurora includes a high-performance storage subsystem. Its MySQL- and PostgreSQL-compatible database engines are customized to take advantage of that fast distributed storage. The underlying storage grows automatically as needed. An Aurora cluster volume can grow to a maximum size of 128 tebibytes (TiB). Aurora also automates and standardizes database clustering and replication, which are typically among the most challenging aspects of database configuration and administration.

Aurora is part of the managed database service Amazon Relational Database Service (Amazon RDS). Amazon RDS is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. If you are not already familiar with Amazon RDS, see the [Amazon Relational Database Service User Guide](#).

The following points illustrate how Amazon Aurora relates to the standard MySQL and PostgreSQL engines available in Amazon RDS:

- You choose Aurora MySQL or Aurora PostgreSQL as the DB engine option when setting up new database servers through Amazon RDS.
- Aurora takes advantage of the familiar Amazon Relational Database Service (Amazon RDS) features for management and administration. Aurora uses the Amazon RDS AWS Management Console interface, AWS CLI commands, and API operations to handle routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair.
- Aurora management operations typically involve entire clusters of database servers that are synchronized through replication, instead of individual database instances. The automatic clustering, replication, and storage allocation make it simple and cost-effective to set up, operate, and scale your largest MySQL and PostgreSQL deployments.
- You can bring data from Amazon RDS for MySQL and Amazon RDS for PostgreSQL into Aurora by creating and restoring snapshots, or by setting up one-way replication. You can use push-button migration tools to convert your existing RDS for MySQL and RDS for PostgreSQL applications to Aurora.

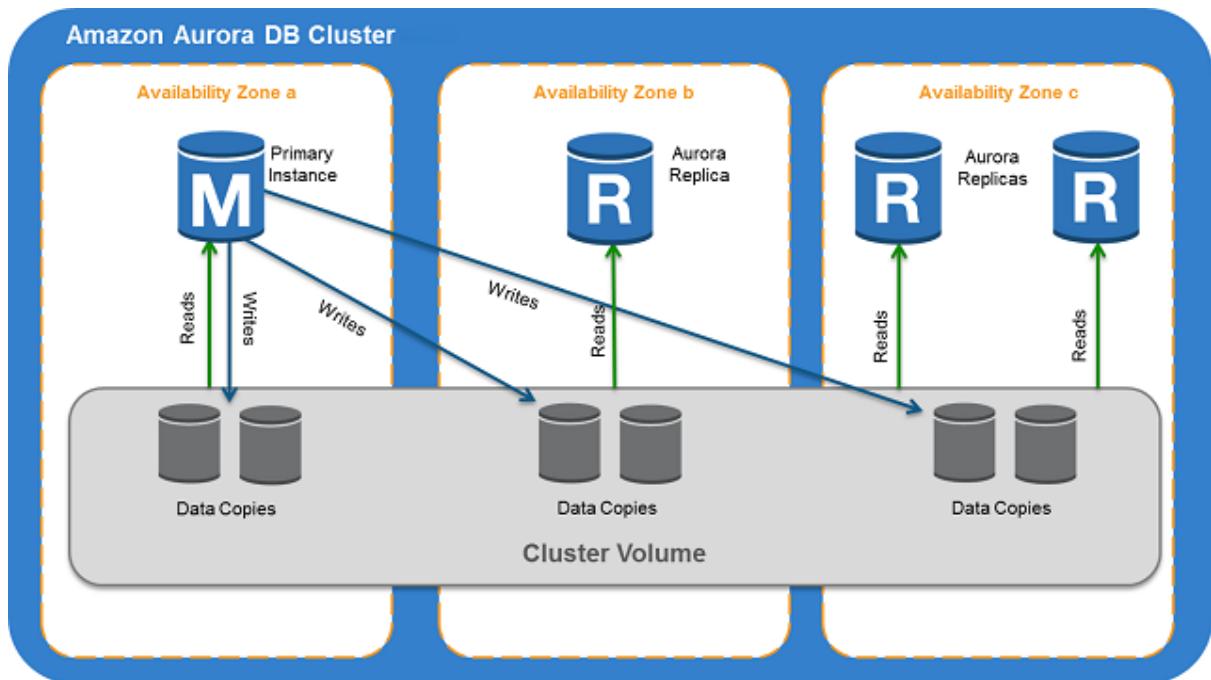
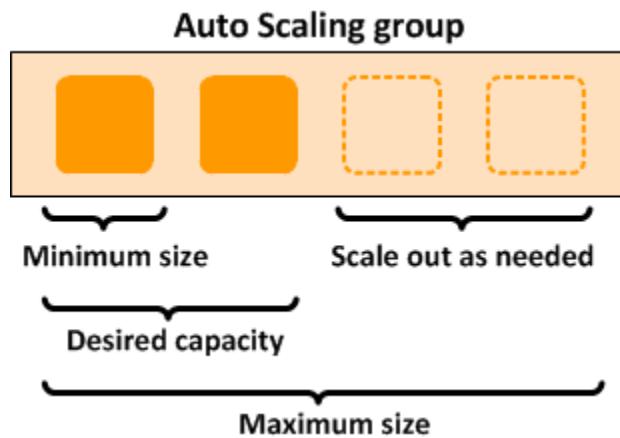


Fig.Amazon Auora DB Clusters

AWS Autoscaling:

Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called Auto Scaling groups. You can specify the minimum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes below this size. You can specify the maximum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes above this size. If you specify the desired capacity, either when you create the group or at any time thereafter, Amazon EC2 Auto Scaling ensures that your group has this many instances. If you specify scaling policies, then Amazon EC2 Auto Scaling can launch or terminate instances as demand on your application increases or decreases.

For example, the following Auto Scaling group has a minimum size of one instance, a desired capacity of two instances, and a maximum size of four instances. The scaling policies that you define adjust the number of instances, within your minimum and maximum number of instances, based on the criteria that you specify.



Auto scaling benefits

Adding Amazon EC2 Auto Scaling to your application architecture is one way to maximize the benefits of the AWS Cloud. When you use Amazon EC2 Auto Scaling, your applications gain the following benefits:

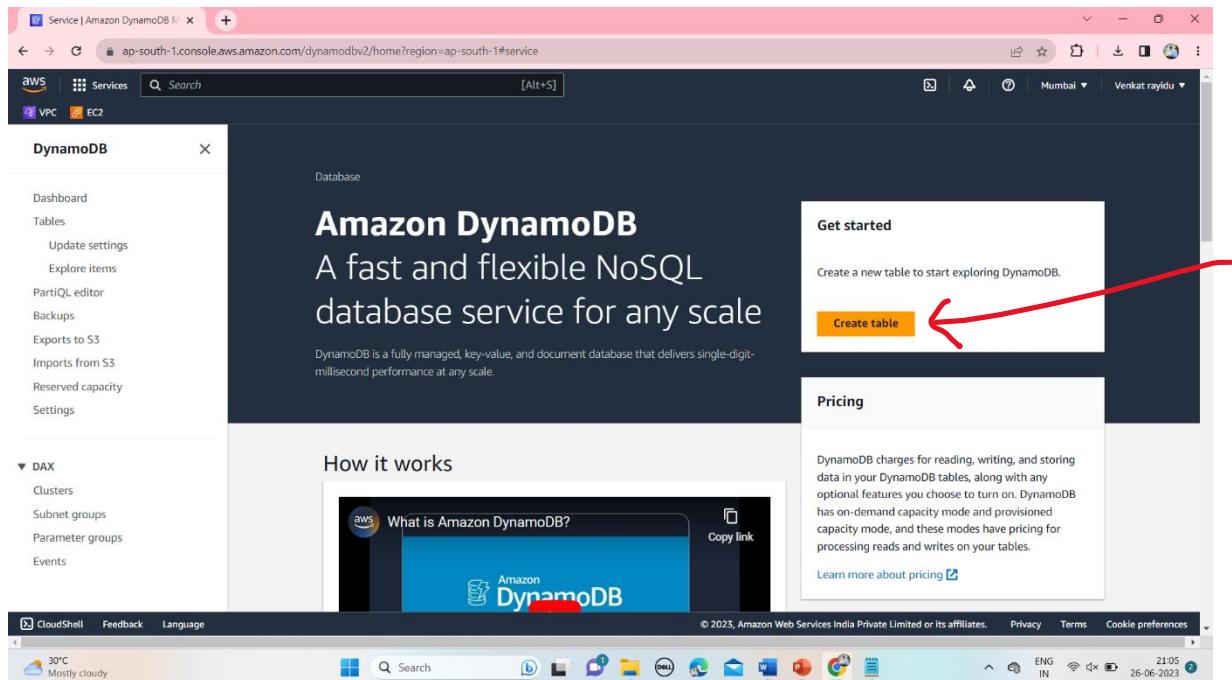
- Better fault tolerance. Amazon EC2 Auto Scaling can detect when an instance is unhealthy, terminate it, and launch an instance to replace it. You can also configure Amazon EC2 Auto Scaling to use multiple Availability Zones. If one Availability Zone becomes unavailable, Amazon EC2 Auto Scaling can launch instances in another one to compensate.
- Better availability. Amazon EC2 Auto Scaling helps ensure that your application always has the right amount of capacity to handle the current traffic demand.
- Better cost management. Amazon EC2 Auto Scaling can dynamically increase and decrease capacity as needed. Because you pay for the EC2 instances you use, you save money by launching instances when they are needed and terminating them when they aren't.

IMPLEMENTATION

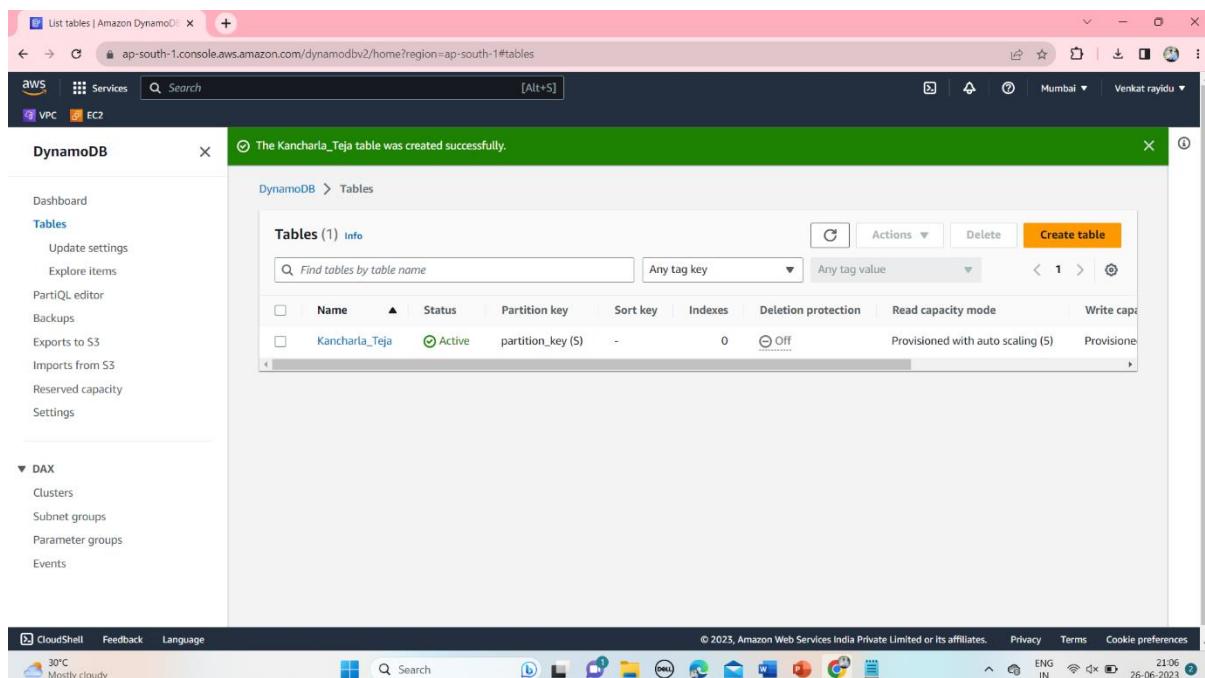
Steps to perform:

DynamoDB

At first we create a DynamoDB.



To create DynamoDB open the above tab and create table



The table "Kancharla_Teja" is created successfully.

The screenshot shows the AWS DynamoDB console. On the left, the navigation menu includes 'Dashboard', 'Tables', 'Update settings', 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', 'Reserved capacity', and 'Settings'. Under 'Tables', there is a sub-menu for 'DAX' with options like 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main content area shows a table named 'Kancharla_Teja' with one item listed. A context menu is open over the item, with 'Create item' highlighted. Other options in the menu include 'Edit capacity', 'Update table class', 'Delete table', 'Create index', 'Create replica', 'Export to S3', 'Turn on TTL', 'Manage tags', and 'Create access control policy'. Below the table, there is a section for 'General information' showing details like 'Partition key: partition_key (String)', 'Sort key: -', 'Capacity mode: Provisioned', and 'Table status: Active'. There is also a note about 'Protect your DynamoDB table from accidental writes and deletes'.

For that table we create items.

The screenshot shows the AWS DynamoDB console after creating two items. A green success message at the top states 'The item has been saved successfully.' The main content area shows the 'Kancharla_Teja' table with two items: 'world' and 'Name'. The 'world' item has a value of 'But iam good'. The 'Name' item has a value of 'Iam Teja'. The 'Items returned' section shows these two items. A message below the items says 'Completed. Read capacity units consumed: 0.5'. The navigation bar at the bottom indicates it's 21:08 on June 26, 2023, in Mumbai, India.

We inserted items called "world" and "Name".

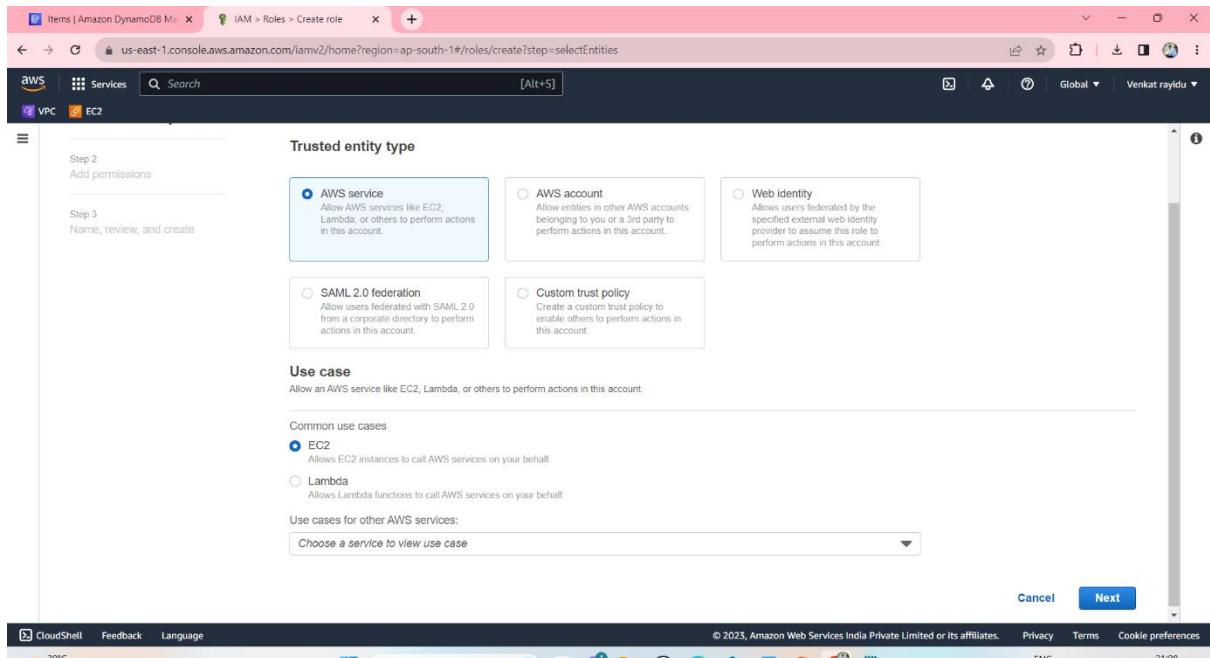
The screenshot shows the AWS IAM Management Console homepage. On the left, there's a navigation sidebar with sections like 'Access management' (User groups, Users, Roles, Policies), 'Identity providers', 'Account settings', 'Access reports' (Access analyzer, Archive rules, Analyzers, Settings, Credential report), and 'CloudShell', 'Feedback', 'Language'. The main content area has a heading 'Identity and Access Management (IAM)'. It displays several status indicators: 'Add MFA for root user' (warning icon), 'Root user has no active access keys' (green checkmark icon), 'IAM resources' (User groups: 0, Users: 0, Roles: 10, Policies: 0, Identity providers: 0), and 'What's new' (with a link to 'View all'). A red arrow points from the top right towards the 'Create role' button in the next screenshot.

Next we want to create “ROLE” called IAM ROLE.

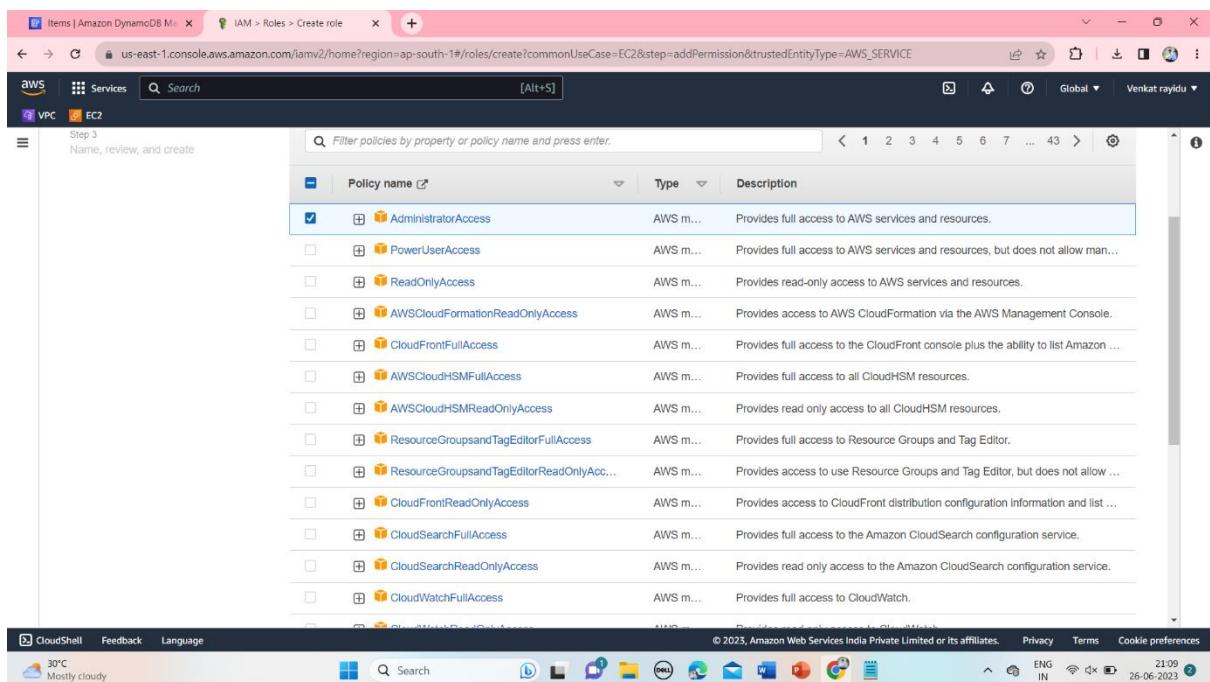
The screenshot shows the 'Roles' page under the 'IAM' section. The sidebar is identical to the previous screenshot. The main area shows a table titled 'Roles (10) Info' with columns for 'Role name', 'Trusted entities', and 'Last act...'. The table lists ten roles, each with a timestamp of when it was last used. A red arrow points from the top right towards the 'Create role' button at the top right of the table.

Role name	Trusted entities	Last act...
AWSCloud9SSMAccessRole	AWS Service: cloud9, and 1 more.	4 days ago
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable	AWS Service: dynamodb.application-autoscaling (Service-Linked Role)	20 minutes ago
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Linked Role)	5 days ago
AWSServiceRoleForAWSCloud9	AWS Service: cloud9 (Service-Linked Role)	4 days ago
AWSServiceRoleForEc2InstanceConnect	AWS Service: ec2-instance-connect (Service-Linked Role)	4 days ago
AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (Service-Linked Role)	5 days ago
AWSServiceRoleForRDS	AWS Service: rds (Service-Linked Role)	1 hour ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)	-
rds-monitoring-role	AWS Service: monitoring.rds	-

For creating Role open the above tab.



Here we selected EC2 case for instance and DynamoDB instance.



For this we give Access called AdministratorFullAccess.

The screenshot shows the AWS IAM Roles page. A green banner at the top indicates that a role has been created. The main table lists 11 roles, each with a checkbox, a role name, a list of trusted entities, and a timestamp. The roles listed are: AWSCloud9SSMAccessRole, AWSServiceRoleForApplicationAutoScaling_DynamoDBTable, AWSServiceRoleForAutoScaling, AWSServiceRoleForAWSCloud9, AWSServiceRoleForEc2InstanceConnect, AWSServiceRoleForElasticLoadBalancing, AWSServiceRoleForRDS, and AWSServiceRoleForSupport.

Now the role is created successfully.

The screenshot shows the AWS IAM Create User page, Step 1: Specify user details. The user name is set to "TEJA_24". The "Provide user access to the AWS Management Console - optional" checkbox is checked. A callout box titled "Are you providing console access to a person?" asks if the user wants to create an IAM user or specify a user in Identity Center. The "I want to create an IAM user" option is selected. Below this, there are fields for "Console password" and "Autogenerated password".

Next we create User and give full administratorAccess.

The screenshot shows the AWS IAM 'Create user' page. A green success message at the top states 'User created successfully'. Below it, a note says 'You can view and download the user's password and email instructions for signing in to the AWS Management Console.' On the left, a vertical navigation bar lists steps: Step 1 (Specify user details), Step 2 (Set permissions), Step 3 (Review and create), and Step 4 (Retrieve password). The current step is Step 4, 'Retrieve password'. To the right, under 'Console sign-in details', there is a 'Console sign-in URL' (https://231769321415.signin.aws.amazon.com/console), a 'User name' (TEJA_24), and a 'Console password' (redacted). Buttons at the bottom include 'Download .csv file' and 'Return to users list'.

The user Teja_24 is created.

The screenshot shows the AWS EC2 'Instances' page. The left sidebar includes links for EC2 Dashboard, EC2 Global View, Events, Limits, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, and AMI Catalog. The main content area is titled 'Instances info' and shows a search bar with 'Find instance by attribute or tag (case-sensitive)' and a filter 'Instance state = running'. A table header includes columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IP. A message below the table says 'No matching instances found'. At the bottom, a modal window titled 'Select an instance' is open.

Next we will create instance

The screenshot shows the AWS EC2 Management Console. In the top navigation bar, there are tabs for IAM > Users > Create user, Items | Amazon DynamoDB M..., and Launch an instance | EC2 Management. The main content area displays a success message: "Successfully initiated launch of instance (i-0c8b1923bf93f04ba)". Below this, there is a "Launch log" button. A "Next Steps" section follows, containing links to "Create billing and free tier usage alerts", "Connect to your instance", "Connect an RDS database", and "Create EBS snapshot policy". A search bar at the top of the steps section asks, "What would you like to do next with this instance, for example 'create alarm' or 'create backup'?" A navigation bar at the bottom includes CloudShell, Feedback, Language, and various system icons.

Instance is created successfully.

The screenshot shows the AWS EC2 Management Console on the Instances page. The left sidebar includes sections for EC2 Dashboard, EC2 Global View, Events, Limits, Instances (selected), Instances Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, and AMI Catalog. The main content area displays a table titled "Instances (1/3) Info" with one row for the instance "teja" (Instance ID: i-0c8b1923bf93f04ba, State: Running). A context menu is open over this instance, showing options like Connect, View details, Manage instance state, Instance settings, Networking, Security (selected), Image and templates, and Monitor and troubleshoot. The security option has a sub-menu with "Modify IAM role" highlighted. Below the table, a detailed view for "Instance: i-0c8b1923bf93f04ba (teja)" is shown, including tabs for Details, Security, Networking, Storage, Status checks, Monitoring, and Tags. The Details tab shows Instance ID (i-0c8b1923bf93f04ba), Public IPv4 address (43.205.103.252), Private IPv4 addresses (172.31.42.37), and Public IPv4 DNS (ec2-43-205-103-252.ap-south-1.compute.amazonaws.com). The security tab shows the IAM role "ec2-43-2". A navigation bar at the bottom includes CloudShell, Feedback, Language, and various system icons.

Modifying IAM Role(connecting Dynamodb with instance).

Now go to Security Credintials

IAM > Users > TEJA_24

Identity and Access Management (IAM)

Summary

ARN: arn:aws:iam::231769321415:user/TEJA_24

Console access: Enabled without MFA

Created: June 26, 2023, 21:10 (UTC+05:30)

Last console sign-in: Never

Access key 1: Not enabled

Access key 2: Not enabled

Permissions | Groups | Tags | **Security credentials** | Access Advisor

Console sign-in

Console sign-in link: https://231769321415.signin.aws.amazon.com/console

Console password: Updated 15 minutes ago (2023-06-26 21:10 GMT+5:30)

Last console sign-in: Never

Click on Security credentials...

Create Access Key

IAM > Users > TEJA_24

Identity and Access Management (IAM)

Device type Identifier Certifications Created on

No MFA devices. Assign an MFA device to improve the security of your AWS environment

Assign MFA device

Access keys (0)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more

Create access key

No access keys

As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. Learn more

Create access key

SSH public keys for AWS CodeCommit (0)

User SSH public keys to authenticate access to AWS CodeCommit repositories. You can have a maximum of five SSH public keys (active or inactive) at a time. Learn more

Actions Upload SSH public key

SSH Key ID	Uploaded	Status
------------	----------	--------

Click on Access Key And click on Command Line Interface and create.

The screenshot shows the AWS IAM 'Create New Access Key' page. A green success banner at the top states: 'Access key created. This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.' Below this, the 'Retrieve access keys' section displays the generated access key (AKIAITL5UHF7DVXJLUIXD) and its corresponding secret access key (XXXXXXXXXXXXXX). A link to 'Access key best practices' is provided, along with a 'Download .csv file' button and a 'Done' button.

Access Key is created successfully.

Here we connected the DynamoDB to Instance with the help of IAM Role and IAM User by giving permissions called Access key.

Lets Check for This Architecture is works or not

Connect The Instance

The screenshot shows the EC2 Instance Connect terminal session. It displays a black terminal window with white text. The text includes the URL 'https://aws.amazon.com/amazon-linux-2/fec2-user@ip-172-31-42-37 ~\$' and a prompt for the user 'i-0c8b1923bf93f04ba (teja)'. Below the terminal, the status bar indicates 'PublicIPs: 43.205.103.252 PrivateIPs: 172.31.42.37'. At the bottom, the Windows taskbar shows various icons and the system tray.

The instance is connected

```

Last login: Mon Jun 26 16:11:40 2023 from ec2-13-233-177-5.ap-south-1.compute.amazonaws.com
[ec2-user@ip-172-31-42-37 ~]$ sudo su
[root@ip-172-31-42-37 ec2-user]# aws dynamodb
Note: AWS CLI version 2, the latest major version of the AWS CLI, is now stable and recommended for general use. For more information, see the AWS CLI version 2 installation instructions at: https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html
usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

aws help
aws <command> help
aws <command> <subcommand> help
aws: error: too few arguments
[root@ip-172-31-42-37 ec2-user]# aws configure
AWS Access Key ID [None]: AKIATL5UHFTDVXJL0IXD
AWS Secret Access Key [None]: Hu2YfL5zxRSarUrTemaDX4Y0fRqx2ReXcgxA57eC1
Default region name [None]: ap-south-1
Default output format [None]: json

i-0c8b1923bf93f04ba (teja)
PublicIPs: 43.205.103.252 PrivateIPs: 172.31.42.37

```

Arguments

- 1.sudo su → is used for giving all admin permissions to normal user
- 2.aws dynamodb →Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
- 3.aws configure
- 4.Copy and paste Access Key
- 5.Copy and paste password
- 6.ap-south-1 → enter region
- 7.aws dynamodb list -tables
- 8.aws dynamodb scan --table -Network

```

Last login: Mon Jun 26 16:11:40 2023 from ec2-13-233-177-5.ap-south-1.compute.amazonaws.com
[ec2-user@ip-172-31-42-37 ~]$ sudo su
[root@ip-172-31-42-37 ec2-user]# aws dynamodb
Note: AWS CLI version 2, the latest major version of the AWS CLI, is now stable and recommended for general use. For more information, see the AWS CLI version 2 installation instructions at: https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html
usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

aws help
aws <command> help
aws <command> <subcommand> help
aws: error: too few arguments
[root@ip-172-31-42-37 ec2-user]# aws configure
AWS Access Key ID [None]: AKIATL5UHFTDVXJL0IXD
AWS Secret Access Key [None]: Hu2YfL5zxRSarUrTemaDX4Y0fRqx2ReXcgxA57eC1
Default region name [None]: ap-south-1
Default output format [None]:
[root@ip-172-31-42-37 ec2-user]# aws dynamodb list-tables
{
    "TableNames": [
        "Kancharia_Teja"
    ]
}
i-0c8b1923bf93f04ba (teja)
PublicIPs: 43.205.103.252 PrivateIPs: 172.31.42.37

```

Finally the table is Displayed.

```

aws | Services | Search [Alt+S]
VPC EC2
"Kancharla_Teja"
}
[root@ip-172-31-42-37 ec2-user]# aws dynamodb scan --table-name Kancharla_Teja
{
  "Count": 2,
  "Items": [
    {
      "partition_key": {
        "S": "Name"
      },
      "NewValue": {
        "S": "Iam Teja"
      }
    },
    {
      "partition_key": {
        "S": "world"
      },
      "NewValue": {
        "S": "But iam good"
      }
    }
  ],
  "ScannedCount": 2,
  "ConsumedCapacity": null
}
[root@ip-172-31-42-37 ec2-user]#
i-Oc8b1923bf93f04ba (teja)
PublicIPs: 43.205.103.252 PrivateIPs: 172.31.42.37

```

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 28°C Partly cloudy ENG IN 21:45 26-06-2023

Finally the items presented in table is displayed.

Now we will upload files by creating S3 Bucket

1.Create S3 Bucket

Create bucket [Info](#)
Buckets are containers for data stored in S3. [Learn more](#)

General configuration

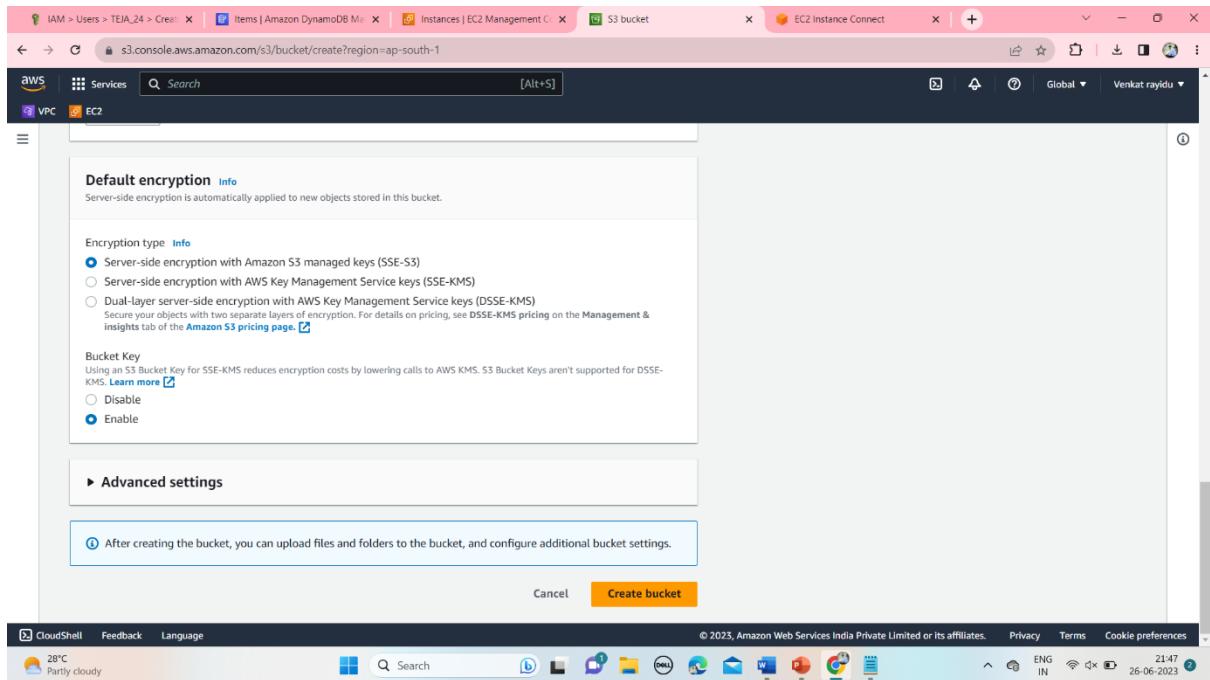
Bucket name
 Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

AWS Region

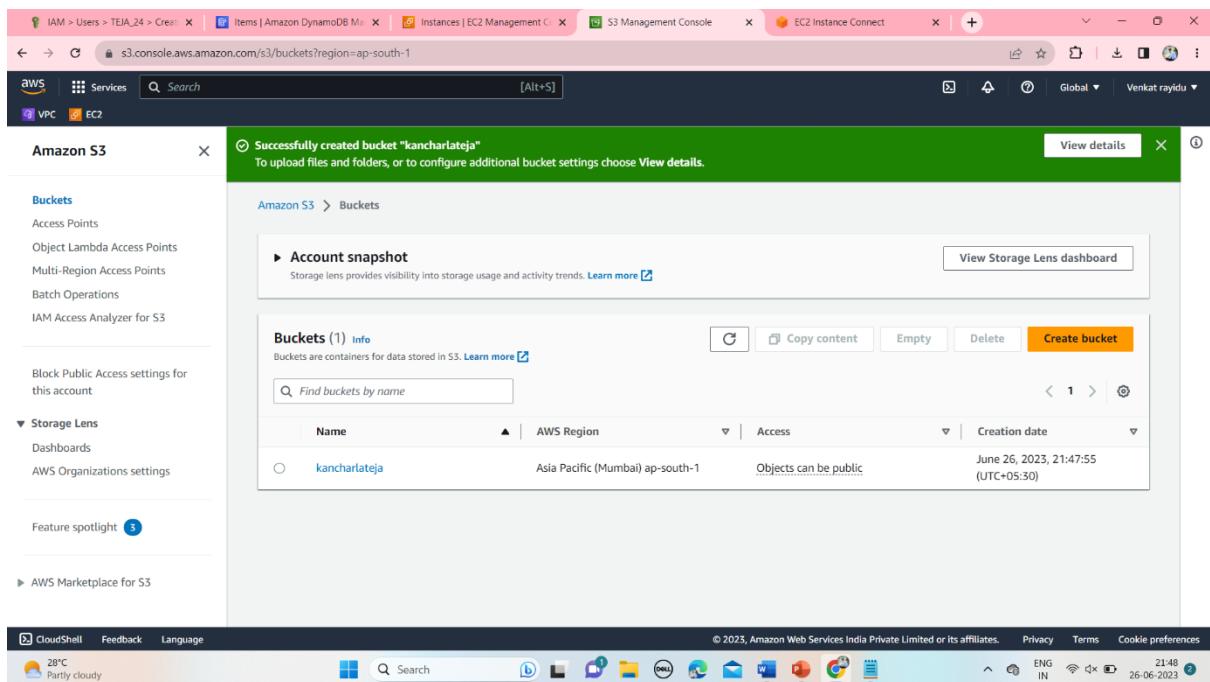
Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.
[Choose bucket](#)

Object Ownership [Info](#)
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 28°C Partly cloudy ENG IN 21:47 26-06-2023



Click on Create Bucket



Finally the bucket was created.

Now we create IAM Role using Lambda function

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Trusted entity type

- AWS service Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account Allows 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 Allows users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy Creates a custom trust policy to enable others to perform actions in this account.

Use case
Allow an AWS service like EC2, Lambda, or others 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:

Here we use Lambda function

Identity and Access Management (IAM)

Role connectingfromlambda created.

Roles (12) An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last act...
AWSCloud9SSMAccessRole	AWS Service: ec2, and 1 more.	4 days ago
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable	AWS Service: dynamodb.application-autoscaling (Service-Linked Role)	30 minutes ago
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Linked Role)	5 days ago
AWSServiceRoleForAWSCloud9	AWS Service: cloud9 (Service-Linked Role)	4 days ago
AWSServiceRoleForEc2InstanceConnect	AWS Service: ec2-instance-connect (Service-Linked Role)	4 days ago
AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (Service-Linked Role)	5 days ago
AWSServiceRoleForRDS	AWS Service: rds (Service-Linked Role)	30 minutes ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-

Here we give full Access to user and Role is created.

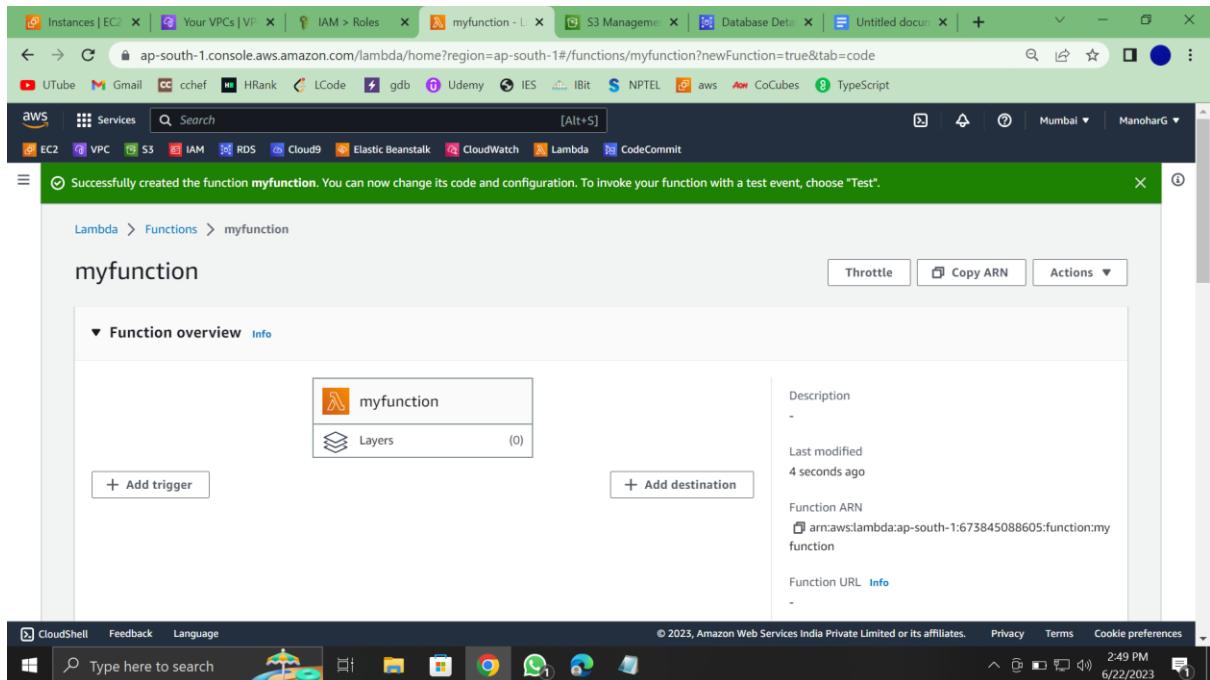
lambda

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The 'Basic information' step is active. It includes fields for 'Function name' (set to 'my functionName'), 'Runtime' (set to 'Node.js 18.x'), and 'Architecture' (set to 'Info'). There are three tabs at the top: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. A note at the top states: 'AWS Serverless Application Repository applications have moved to [Create application](#)'.

Now go to lambda and create the function by giving the function name and the runtime
Language like python, ruby and Node.js

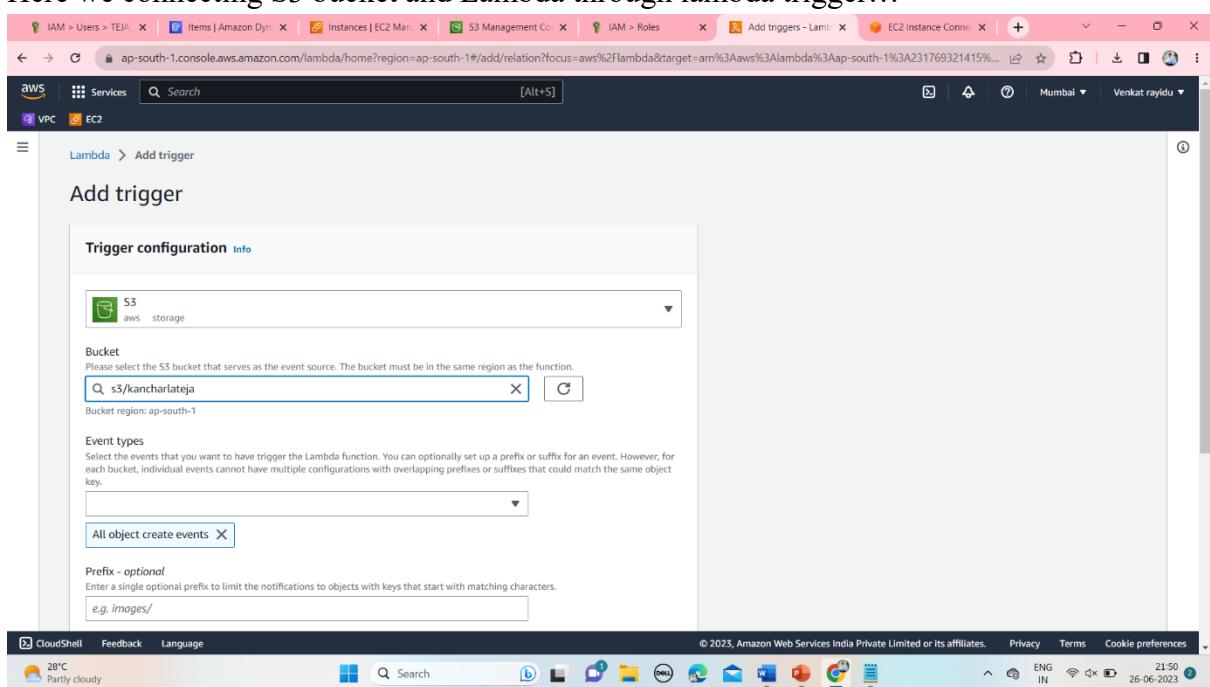
The screenshot shows the 'Create function' wizard in the AWS Lambda console, specifically the 'Permissions' step. It shows the 'Change default execution role' section, where 'Use an existing role' is selected and 'role1' is chosen. Below this, there's a note about CloudWatch Logs permissions. At the bottom right are 'Cancel' and 'Create function' buttons.

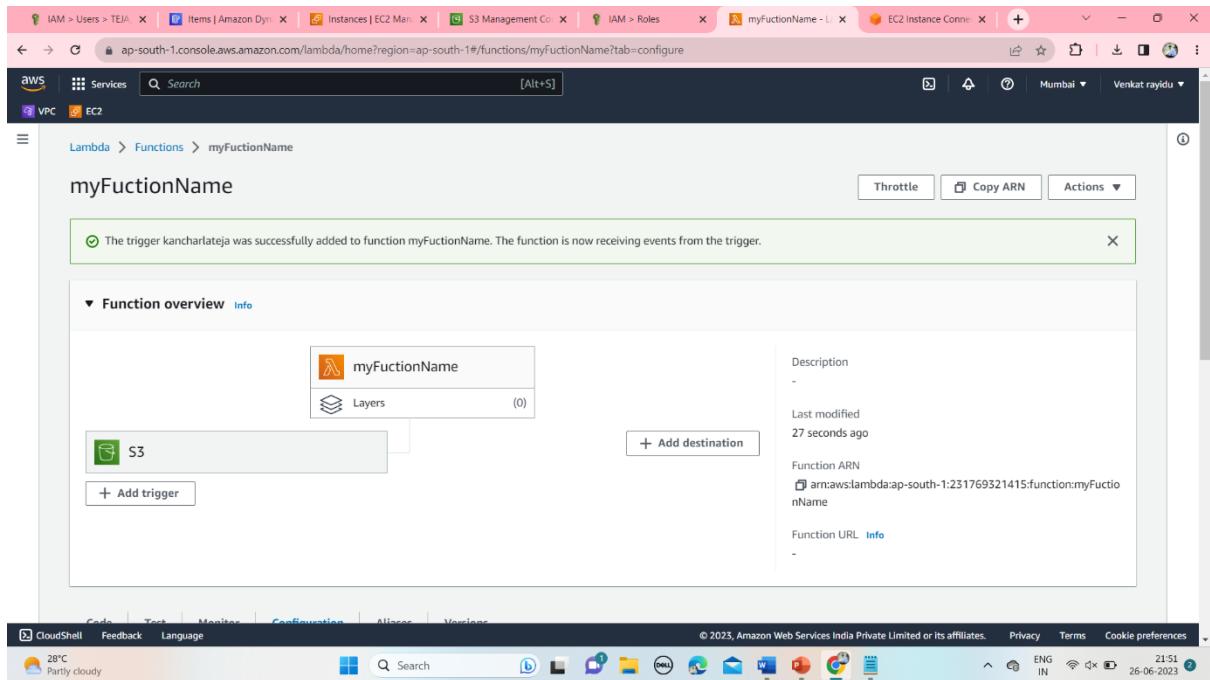
Use the created role (role1) while creating the function and click on create function.



Lambda function is created successfully.

Here we connecting S3 bucket and Lambda through lambda trigger...





Here we see The s3 bucket is connected to lambda

Now We need Code to execute this architecture

```
import boto3
from uuid import uuid4
def lambda_handler(event, context):
    s3 = boto3.client("s3")
    dynamodb = boto3.resource('dynamodb')
    for record in event['Records']:
        bucket_name = record['s3']['bucket']['name']
        object_key = record['s3']['object']['key']
        size = record['s3']['object'].get('size', -1)
        event_name = record['eventName']
        event_time = record['eventTime']
        dynamoTable = dynamodb.Table('TableName')
        dynamoTable.put_item(
            Item={'PartitionKeyName': str(uuid4()), 'Bucket': bucket_name, 'Object': object_key, 'Size': size, 'Event': event_name, 'EventTime': event_time})
```

```

1 import boto3
2 from uuid import uuid4
3 def lambda_handler(event, context):
4     s3 = boto3.client('s3')
5     dynamodb = boto3.resource('dynamodb')
6     for record in event['Records']:
7         bucket_name = record['s3']['bucket']['name']
8         object_key = record['s3']['object']['key']
9         size = len(record['s3']['object']['size'])
10        event_name = record['eventName']
11        event_time = record['eventTime']
12        dynamoTable = dynamodb.Table("Kancharla_Teja")
13        dynamoTable.put_item(
14            Item={"partition_key": str(uuid4()), 'Bucket': bucket_name, 'Object': object_key, 'Size': size, 'Event': event_name, 'EventTime': event_time})

```

The screenshot shows the AWS Lambda function editor with the code for the function 'myFuctionName'. The code uses the boto3 library to interact with S3 and DynamoDB. It iterates over records from an S3 event, extracts bucket name, object key, size, and event details, and then inserts this information into a DynamoDB table named 'Kancharla_Teja'.

Deploy the code means execute the code

The screenshot shows the AWS S3 console. On the left, there's a sidebar with 'Amazon S3' navigation. Under 'Buckets', it lists 'Access Points', 'Object Lambda Access Points', 'Multi-Region Access Points', 'Batch Operations', and 'IAM Access Analyzer for S3'. Under 'Storage Lens', it lists 'Dashboards' and 'AWS Organizations settings'. There's also a 'Feature spotlight' section and a link to 'AWS Marketplace for S3'. The main area shows the 'kancharlateja' bucket under 'Buckets'. The 'Objects' tab is selected, showing '0 Objects'. A message says 'No objects' and 'You don't have any objects in this bucket.' There are buttons for 'Upload' and 'Actions'. Below the table, there's a search bar and a 'Show versions' checkbox.

Now upload the any file into S3 Bucket.

The screenshot shows the AWS S3 Management Console with a green success banner at the top stating "Upload succeeded". Below it, a summary table shows one file uploaded to the destination "s3://kancharlateja" with a status of "Succeeded" (1 file, 668.0 B (100.00%)). There are tabs for "Files and folders" and "Configuration". A detailed list of files is shown below, including "iamdba+trigger.txt" from the "Files and folders" tab.

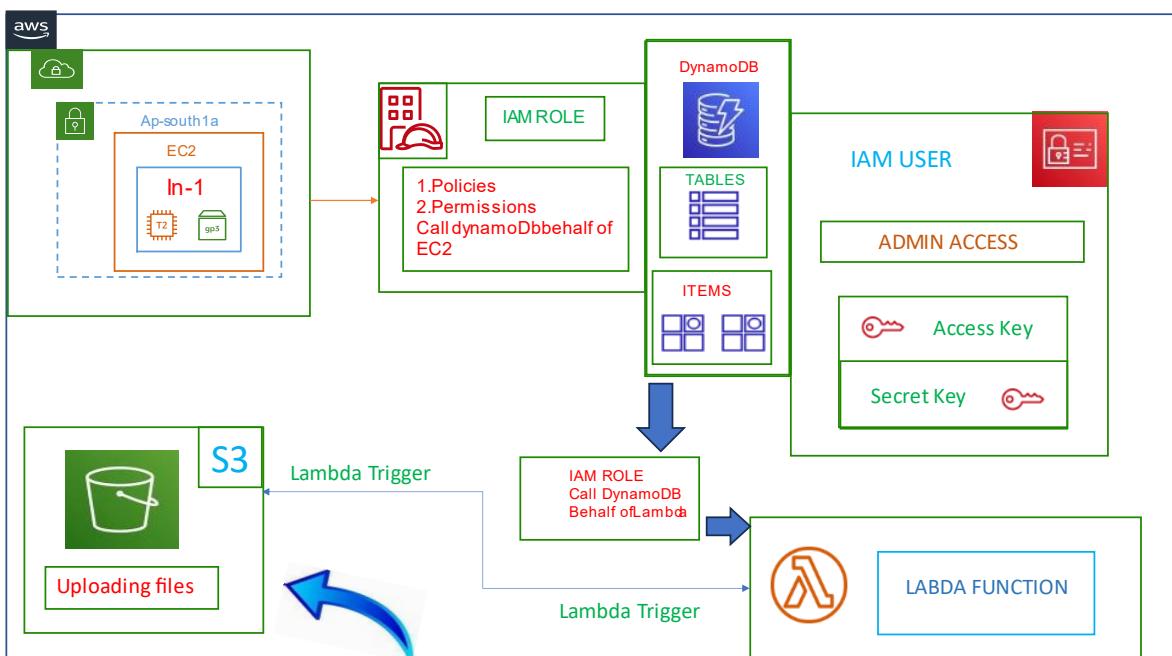
Now the file will uploaded successfully.

Now we check the files is uploaded or not

The screenshot shows the AWS CloudShell interface displaying the contents of the uploaded file "iamdba+trigger.txt". The file contains JSON-like data with fields such as "S": "Iam Teja", "EventTime": "2023-06-26T16:23:42.489Z", "Object": "iamdba+trigger.txt", "Bucket": "kancharlateja", and "partition_key": "60065be5-bf3d-413a-85c4-6f4f71cb723d". The CloudShell also shows the command history and environment variables.

Here we see The uploaded files here Hence Our Architecture executed Successfully.

Finally the file was uploaded successfully into dynamodb.



Finally The Above Architecture executed successfully.

Conclusion:

In conclusion, the project successfully implemented an automated file transfer solution from Amazon S3 to DynamoDB using AWS Lambda and IAM roles. By creating an IAM role with appropriate permissions, configuring a Lambda function triggered by S3 events, and writing code to handle the file transfer process, the project achieved seamless integration between the two services. The automation streamlines the data pipeline, enhances efficiency, reduces manual intervention, and minimizes errors. This project serves as a reliable and scalable solution for transferring files from S3 to DynamoDB, facilitating data synchronization and enabling efficient data migration and integration.