**AWS Certified Developer Associate**

* **AWS Accounts**

aws.amazon.com/free

Account Root User:

email = zarod2019@gmail.com

pwd = Serafines@2024

alias = zar-training

account id = 436488467655

IAM User:

user = Zarbio

pwd = Serafines@2024

alias = zar-training

IAM User:

user = Joe

pwd = Joe@2024

* **Courses**
* AWS Certified **Developer Associate** Exam Training DVA-C02

<https://tcsglobal.udemy.com/course/aws-certified-developer-associate-exam-training/learn/lecture/35900716#overview>

<https://digitalcloud.training/aws-developer-associate-resources/>

* Resources:

<https://digitalcloud.training/aws-certified-developer-course-downloads/>

code, folder aws-dva-code: <https://github.com/nealdct/aws-dva-code/tree/main>

**AWS+Certified+Developer+Associate+Slides.pdf**

* **Notes**
* To build and deploy cloud apps use:

AWS Software Development Kits (AWS SDKs for Python, .Net, and Java)

Command Line Interface (AWS CLI)

AWS Management Console

* Amazon Web Services (AWS):

For General cloud engineering (DevOps), enterprise applications, web solutions.

Used for Startups and Web Applications.

Market leader in cloud.

* Jobs:

AWS Certified Developer

* Certifications:

Amazon Web Services Cloud Practitioner – 2023

valid 3 years

cost $100

AWS Certified Solutions Architect – Associate: for general cloud knowledge

AWS Certified **Developer** – **Associate**: for development

AWS Certified DevOps Engineer – Professional: for DevOps roles

* **AWS Certified Developer – Associate Exam**

A close-up of a certificate

Description automatically generated

[**https://aws.amazon.com/certification/certified-developer-associate/**](https://aws.amazon.com/certification/certified-developer-associate/)

A screenshot of a survey

Description automatically generated

AWS-Certified-Developer-Associate\_Exam-Guide.pdf [**https://d1.awsstatic.com/training-and-certification/docs-dev-associate/AWS-Certified-Developer-Associate\_Exam-Guide.pdf**](https://d1.awsstatic.com/training-and-certification/docs-dev-associate/AWS-Certified-Developer-Associate_Exam-Guide.pdf)

Score of 100 - 1000, minimum passing score is 720

**A close up of text

Description automatically generated**

Last version is **DVA-C02** (57 services). Instead DVA-C01 (33 services).

* TDVA-C02 Exam:

Domain 1: Development with AWS Services

Develop code for applications hosted on AWS

Develop code for AWS Lambda

Use data stores in application development

Identify components and resources for security

Domain 2: Security

Implement authentication and/or authorization for applications and AWS services

Implement encryption by using AWS services

Manage sensitive data in application code

Domain 3: Deployment

Prepare application artifacts to be deployed to AWS

Test applications in development environment

Automate deployment testing

Deploy code by using AWS CI/CD services

Domain 4: Troubleshooting and Optimization  
 Assist in a root cause analysis  
 Instrument code for observability  
 Optimize applications by using AWS services and features

* AWS Service Names: <https://aws.amazon.com/certification/policies/general-information/#AWS_Service_Names>

AWS cheat sheets: <https://tutorialsdojo.com/aws-cheat-sheets/>

[**https://medium.com/@meghanaharishankara/how-to-get-aws-developer-associate-certified-in-just-5-weeks-26c022b4b142**](https://medium.com/@meghanaharishankara/how-to-get-aws-developer-associate-certified-in-just-5-weeks-26c022b4b142)

* Exams, courses, simulator:

**Jon Bonso’s practice exams**: <https://www.udemy.com/course/aws-certified-developer-associate-practice-exams-amazon-dva-c01/?couponCode=BFCPSALE24>

**AWS Certified Developer Associate Exam Training DVA-C02**

[**https://tcsglobal.udemy.com/course/aws-certified-developer-associate-exam-training/learn/lecture/35900716#overview**](https://tcsglobal.udemy.com/course/aws-certified-developer-associate-exam-training/learn/lecture/35900716#overview)

AWS Certified Developer Associate Practice Exams DVA-C02

[**https://tcsglobal.udemy.com/course/aws-developer-associate-practice-exams/learn/quiz/4852736#overview**](https://tcsglobal.udemy.com/course/aws-developer-associate-practice-exams/learn/quiz/4852736#overview)

<https://digitalcloud.training/aws-developer-associate-resources/>

Practice Exams | AWS Certified Developer Associate 2024

<https://tcsglobal.udemy.com/course/aws-certified-developer-associate-practice-tests-dva-c01/learn/quiz/4540356#overview>

AWS Certified Developer Associate DVA-C01 Exam Questions

<https://tcsglobal.udemy.com/course/simulado-amazon-aws-certified-developer-associate-2020/learn/quiz/4788484#overview>

* **AWS Accounts and IAM**
* AWS Free Tier account vs Sandbox

|  |  |
| --- | --- |
| AWS Free Tier | Sandbox / Challenge Labs |
| Create your own AWS free tier account | AWS account is hosted by a provider |
| Full control | Limited control |
| You’re responsible for bills, but we will operate in the free tier and set a billing alarm. | No cloud bills (no risk) |
| For Hands-On Lessons (HOL) | Scenario-based challenges. |

* AWS Account

Need:

Credit card

Dynamic email alias account: john {account alias 1} @gmail.com

Phone to receive SMS verification code

Account Root User: super account

* Create your AWS Free Tier Account:

12 months free

1. url: AWS Free Tier

aws.amazon.com/free

[https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=\*all&awsf.Free%20Tier%20Categories=\*all](https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all)

email = [zarod2019@gmail.com](mailto:zarod2019@gmail.com)

pwd = Serafines@2024

Sign in as a Root user

1. Sign in to the Console
2. Account Configuration and Create a Budget

Configure Account Alias: for IAM

Create Account Alias: ex: dct-tab-training, **zar-training** (created)

A screenshot of a web page

Description automatically generated

Enable access to billing for IAM users:

IAM Dashboard

A close up of a sign

Description automatically generated

Update billing preferences:

option Billing Preferences

A screenshot of a email

Description automatically generated

Create a budget and alarm:

option Budget, $5

User a template (simplified)

Monthly cost budget

Enter your budgeted amount ($): 5.00

option Cost Explorer

* AWS Identity and Access Management (IAM):

Used for **authentication** and **authorization**

Ways to manage AWS:

Console

Command Line Interface - CLI

API through SDKs

1. Authentication: login
2. **Authorization**: **allow** or **deny** access to resources

**Policy**: define what we are **allowed to do**

ex:

Performing API actions like run instances on EC2 that launches a virtual server

GetBucket: retrieves information about buckets

CreateUser: create a user in IAM

Core components of IAM:

Users

User groups: for adding users and applying permissions policies

Roles

Policy

IAM Users:

Best practice not to use the root user account, you must set a strong password and enable Multi-Factor Authentication (**MFA**), then we can create user accounts.

ARN (Amazon Resource Name):

A close up of a sign

Description automatically generated

* Creating IAM Users and Groups:

Create Group:

Policy: AdministratorAccess

Create User:

A screenshot of a computer

Description automatically generated

Because I don’t need single sign on in many cases.

**user = Zarbio**

**pwd = Serafines@2024**

**A screenshot of a login screen

Description automatically generated**

* IAM Authentication and MFA:

MFA:

Signing in with MFA requires an authentication code from an MFA device

Something you have: phone, token device

Something you are: retina scans, fingerprints

Used for Root account and individual IAM user accounts

* Setup Multi-Factor Authentication (MFA) for IAM User account:

2 types of users:

Root user

IAM user

Option User, tab Security credentials, Assign MFA device:

Authenticator app: Google Authenticator, AuthyPhone

* AWS Security Token Service - STS:

Provided temporary credentials.

Permissions Policy: allowed or denied to this specific entity

* Access Control Methods – RBAC & ABAC:

RBAC (Role-Based Access Control):

Job function policies = AWS managed policies

Policy documents can be pre-created by AWS

ABAC (Attribute-Based Access Control):

Using attributes, tags to define access to our resources

Permissions policy

Action = API action (RebootDBInstance, StartDBInstance, StopDBInstance)

* Switching IAM Roles:

Create an IAM role

user = Joe

pwd = Joe@2024

**EC2**:

Run **virtual servers in the Cloud** running Linux, Windows, Mac Os

Create Role:

AmazonEC2FullAccess: launch virtual servers in the cloud, permissions for load balancing EC2

Add permissions:

Users, Permissions policies, Add permissions, Create inline policy, JSON

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "Statement1",

"Effect": "Allow",

"Action": [],

"Resource": []

}

]

}

* **AWS Command Line Interface (CLI)**

Tools: AWS CLI, VS Code and Git

* Install the AWS Command Line Interface (CLI):

Install on a Linux/Windows instance

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

In Windows open CMD:

commands:

aws

aws --version

aws s3 ls // don’t allowed

* Configure **Credentials** for the AWS CLI:

Configure a terminal for a user, to create an **access key** (long term credentials) for AWS CLI

Access key = Access key ID + Secret access key

In Windows Power Shell:

command:

aws configure

aws s3 ls // list buckets

A computer screen with white text

Description automatically generated

AKIAWLIF5OTDZ3R6FXUI

4P4Sb97Ct/ry6Nc21bVQgTp9Lv9SovQcnnLymz44

A black background with white text

Description automatically generated

Doesn’t exist any buckets, no errors

cat .aws/config // contents of the file

response:

region = us-east-1

cat .aws/credentials

response:

aws\_acess\_key\_id =

aws\_secret\_access\_key =

A screenshot of a computer

Description automatically generated

* Overview of Using the AWS CLI:

command:

aws help

aws ec2 help

aws ec2 describe-instances // show status of instances

aws s3 help

aws s3 mb s3://{bucket-name} // mb = make/create **bucket** (**container to upload files**)

aws s3 mb s3://mytestbucket43243jd33x

aws s3 ls // list buckets

Create and upload a file to a bucket:

command:

touch testfile.txt // linux, create a file

New-Item -Path "testfile.txt" -ItemType File -Force // windows, create a file

ls // list directories

aws s3 cp testfile.txt s3://{bucket-name} // cp = copy, upload file to a bucket

aws s3 ls s3://{bucket-name} // show content of file

aws s3 rb s3://{bucket-name} --force // remove bucket with content

aws s3 ls // list buckets

* Assuming IAM Roles (CLI):

Create IAM roles from CLI console

option Roles, EC2-Full-Access

Add profile:

A screenshot of a computer

Description automatically generated

Use arn of the role (don’t use access keys, don’t use credentials file)

command:

aws ec2 describe-instances

aws ec2 describe-instances --profile ec2-full-access // show status of instances

option Roles, S3FullAccess

command:

aws s3 ls // access denied

aws s3 ls --profile ec2-full-access

Create my own profile:

command:

aws configure --profile {name}

add access-key

aws configure --profile {name}

See config and credentials files in computer

* **Amazon VPC, EC2, and ELB**
* Amazon VPC, Security Groups, and NACLs

**Public subnets** are accessible to the outside world (**internet**). **Private** **subnets** **not**.

We control the **VPC router** using the **route table**

**NAT** instances and NAT gateways allow your instances in **private** **subnets** which only have private IP addresses to be able to connect to the internet.

Security Groups and Network ACLs: are firewalls to protect the network traffic that’s able to connect to our EC2 instances.

NACLs don’t apply to traffic within the subnet, is only the ingress and egress traffic.

Security Group are applied to the Elastic Network Interfaces that are attached to each of your EC2 instances.

Security Groups support allow rules only. **Stateful** (if traffic is outbound then any return traffic is the same).

Network ACLs also support deny rules. **Stateless** (need separate rule for outbound and the return traffic inbound)

* Amazon EC2 Overview:

EC2: run virtual servers in the cloud, IaaS (AWS manage the physical hardware and software (virtualization))

igw = Internet Gateway

* Create a Custom VPC:

VPC Console, option Your VPCs, Create VPC

option Subnets, Route tables

GitHub-Code/code/amazon-vpc/custom-vpc.md/Create Public and Private Subnets

EC2 Dashboard, in Network setting choose your VPC

* Amazon EBS and Instance Stores:

EBS = Elastic Block Store:

Block based storage system, not a File-based storage system

Have volumes

AZ = Availability Zone

**Instance Store volumes** are **ephemeral** (data is **non-persistent**). If the power is lost to this host server, all that data is gone forever.

Used for temporary data that can be recreated.

**EBS volumes** offer **persistent** storage. If you have long term data storage, you cannot afford to potentially lose your data.

Amazon EBS Snapshots: to do backups

AMI – Amazon Machine **Image** are backed by a snapshot, is a template contains software configuration (operating system, app server, apps) required to launch your instance. Snapshot ID

* Create and Attach an EBS Volume:

aws-dva-code\amazon-ebs\amazon-ebs-volumes.md

Snapshots are not stored in the availability zone, they’re store on Amazon S3 (regional service).

VPC -> ECS -> Instances -> Volumes, Snapshots

* Amazon Elastic File System (EFS):

Is shared file system, connect instances from multiple availability zones

NFS = Network File System: connection protocol

* Create an Amazon EFS Filesystem:

aws-dva-code\amazon-efs\working-with-efs.md

Create a File system using the Amazon elastic File system service.

EF2 = Managed File Storage for EC2

* Amazon EC2 User Data and Metadata:

**Instance metadata** = data about your EC2 instance

**User data** = way to run scripts when we start our instances the first time

**Metadata** = way that we can retrieve specific information about the instance itself

* Using User Data and Metadata:

aws-dva-code\amazon-ec2\user-data-metadata.md

* Access Keys and IAM Roles with EC2:

Using IAM Role is more secure than using IAM User. IAM Roles use the AWS Security Token Service - STS.

* Practice with Access Keys and IAM Roles:
* **Summary**

NACLs = Network Access Control Lists

CIDR = Classless Inter-Domain Routing, ex. 10.0.0.0**/16**

**NAT** = Network Address Translation, when the instance with the private address wants to connect to the Internet, it will need a public address that can access the Internet.

**Role**: contains **permissions**/**policies** with **credentials**

Each **Profile** with **credential**

KMS = Key Manager Service

To encrypt the password of environment variables

IOPS = Input Output per second = performance for the disc

Use SQS FIFO queues for ordered processing

**User pool** can be used to **authenticate** but the **identity pool** is used to provide **authorized** access to AWS services.

Idempotent = performing it again (using the same inputs) does not change the result.

A diagram of a cloud computing system

Description automatically generated

reduce **cold starts**: latency when service initializes resources for the **first time**

NAT gateway: to enable outbound internet access for resources in private subnets

* **Compute**
* **Amazon Elastic Compute Cloud - EC2**:

ecs-cli: Amazon Elastic Container Service

ec2 instance = terminal instance

**Web service** that provides **resizable compute capacity** in the cloud

Launch **virtual server** instances

Amazon Machine Images (**AMIs**) are templates for your instances (OS (Linux, Ubuntu, Windows Server, MacOS), software packages)

* **AWS Batch:**

Run batch jobs (for ML model training, simulation, analysis) at any scale, in Amazon ECS, EKS, Fargate

* **AWS Elastic Beanstalk:**

Run, deploy and manage **web apps**

Handles the deployment details of capacity provisioning, **load balancing**, **auto-scaling**, application **health** monitoring.

PaaS

Deployment options:

All at once: quickest deployment

Rolling

Rolling with additional batch

Immutable

Blue/Green deployment

* **AWS Lambda:**

Run code in response to events, without thinking about servers or clusters

Function can access:

AWS services or non-AWS services

AWS services running in VPCs (RedShift, Elasticache, RDS instances)

Non-AWS services running on EC2 instances in an AWS VPC

Access to local storage in the /tmp directory

Write and upload code as a .zip file or container image.

Use **environment variables** to adjust your function's behavior **without** updating **code**

Lambda **layer**: .zip file contains supplementary code or data (library dependencies, custom runtime, configuration files)

Invoking a Lambda function asynchronously:

Amazon Simple Storage Service (Amazon S3) and Amazon Simple Notification Service (Amazon SNS) invoke functions asynchronously to process events.

You can configure how Lambda **handles errors**, and can send invocation records to a downstream resource such as Amazon Simple Queue Service (**Amazon SQS**) or Amazon **EventBridge** to chain together components of your application.

A diagram of a diagram

Description automatically generated with medium confidence

* **Container**
* **Amazon Elastic Container - ECS**:

Container Management Service, highly secure, reliable, and scalable way to run containers

run applications on a managed cluster of Amazon EC2 instances

Supports:

**Docker** containers

**Task placement strategies**: algorithm for selecting instances for task placement/termination.

binpack: place tasks based on the **least available amount** of **CPU** or memory. **Minimizes** the number of **instances** in use.

random: place tasks randomly

spread: place tasks based on the specified value.

+AWS Fargate serverless compute for containers

* **Amazon Elastic Container Service for Kubernetes (EKS):**

run, and scale **Kubernetes** without thinking about **cluster** management

**Amazon ECS vs Amazon EKS**

* **AWS Copilot:**

deploying and managing **containerized** applications on AWS (ECS, AWS Fargate)

automate deployments using **AWS CodePipeline**, which integrates with services like **AWS CodeCommit**, **AWS CodeBuild**, and **AWS CodeDeploy**.

Infrastructure-as-code (IaC) templates

A screenshot of a computer

Description automatically generated

* **Amazon Fargate:**

Serverless compute engine for containers for ECS and EKS

Manage your **applications**, not infrastructure

Improve security through isolation for Amazon ECS tasks and Amazon EKS pods.

A screenshot of a computer

Description automatically generated

* **Analytics**
* **Amazon Athena:**

Query data in S3 using SQL, for data stored in relational, non-relational, object, custom data sources.

**Amazon Athena <–> S3**

* **Amazon CloudSearch -> Amazon OpenSearch Service:**

Managed Search service

* **Amazon Kinesis:**

Analyze **real-time** **video** and data **streams**

Uses:

Real-time apps (monitoring, fraud detection, live leaderboards)

From batch to real-time analytics

Analyze IoT device data: to send real-time alerts, thresholds

Video analytics apps: video playback, security monitoring, face detection, ML

* **Amazon Kinesis Data Firehose:**

Real-time streaming delivery for any data, at any scale, at low-cost

Uses:

Data Transformation: invoke your **Lambda** function to **transform** incoming source data and deliver the **transformed data** to destinations (S3, HTTP endpoint)

* **Application Integration**
* **AWS Step Functions:**

**Visual** workflows for distributed applications

Uses:

Automate workflows without code

Orchestrate microservices

Create data and machine learning (ML) pipelines

* **Amazon API Gateway:**

Create, publish, maintain, monitor and secure REST APIs and WebSocket APIs.

supports mock integrations for API methods.

Authorize access to your APIs with AWS Identity, Access Management (IAM), Amazon Cognito

API Gateway Stage Variables: environment configuration (dev, test, prod).

* **Amazon EventBridge:**

Serverless event bus for SaaS apps & AWS services

Uses:

event-driven architectures

create point-to-point integrations

* **Amazon Simple Notification Service - SNS:**

Fully managed messaging service

Exchange of messages between **distributed systems**, **microservices** or applications using **Pub/sub** (publish/subscribe), **SMS**, **email** and **mobile** **push** **notifications**

FIFO messaging

Encrypt messages with AWS Key Management Service (KMS)

* **Amazon Simple Queue Service - SQS:**

Managed message queues for microservices, distributed systems, serverless apps

FIFO queues

Security with AWS Key Management (KMS), HTTPS/TLS

* **AWS AppSync:**

Fully-managed, scalable **GraphQL APIs**

Connect applications to events, data, AI models

to access data from **multiple databases**, **micro-services**, and AI models with a **single** GraphQL API request

GraphQL for **DynamoDB** and **Aurora** databases

* **Database**
* **Amazon Aurora:**

Manage **relational** database for MySQL and PostgreSQL

has 5x the throughput of MySQL and 3x of PostgreSQL

data durable across 3 AZs (customers only pay for 1 copy)

* **Amazon DynamoDB:**

Managed **NoSQL** database

key-value and document database

serverless

data synchronously replicated across 3 AZs in a region

DynamoDB **Streams**:

build serverless **event-driven** applications

captures a time-ordered sequence of item-level modifications in any DynamoDB table and stores this information in a log for up to 24 hours.

+AWS Lambda triggers: code automatically respond to events in DynamoDB Streams

DynamoDB **DAX**: increase **performance** of DynamoDB tables and offload read requests. **Cannot** used in front of an Amazon **RDS** database.

* **Amazon ElastiCache:**

**In-memory** **database** **cache** used in front of Amazon **RDS**

web service to deploy and run **Memcached** or **Redis** server nodes (replication) in the cloud.

Uses:

For **encryption** and **high availability** use ElastiCache **Redis** with **cluster mode** enabled. **Memcached** does **not** support encryption or high availability.

Lowest latency

Real-time application **data caching: Real-time performance** for **real-time applications**

Real-time session stores: session data for gaming, e-commerce, social media, online apps

A screenshot of a computer

Description automatically generated

* **Amazon RDS - Relational Database Service:**

Managed relational database service for PostgreSQL, MySQL, MariaDB, SQL Server, Oracle, Db2

Amazon Aurora: MySQL, PostgreSQL

* **Developer Tools**

code, build, test and deploy app

SDKs, code editors, CI/CD

* **AWS Cloud9:**

Write, run, and debug code on a cloud IDE

* **AWS CloudShell:**

Browser-based shell environment

AWS CLI in browser, Linux

<https://aws.amazon.com/cloudshell/?nc2=h_ql_prod_dt_cs>

* **AWS Command Line Interface - CLI:**

Unified tool to manage AWS services

<https://aws.amazon.com/cli/>

<https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-sso.html>

* **AWS CodeBuild:**

Build and test code

* **AWS CodeCommit:**

Managed version control service

host private Git repositories

Store code in private Git repositories

* **AWS CodeDeploy:**

Automate code deployments

2 deployment type:

in-place

blue/green: AWS Lambda, Amazon ECS

* **AWS CodePipeline:**

Delivery pipeline to deliver application

Release software using continuous delivery

Is triggered by changes to the main branch of an AWS **CodeCommit** repository

Use AWS **CodeBuild** to **test** and **build**, and AWS **CodeDeploy** to **deploy** the app

* **AWS X-Ray:**

Analyze and **debug** your applications

Analyze the behavior of their production, distributed applications with **end-to-end tracing capabilities**. To identify performance **bottlenecks**, **edge case errors**, **detect issues**.

* **Management & Governance**
* **Amazon CloudWatch:**

Monitoring service for AWS cloud resources and applications

Monitor resources and applications on AWS, on premises, and on other clouds

Used to trigger actions based on changes in the state of AWS services, collect and track **metrics**, collect and monitor **log files**, and set **alarms**. Provides **insights** into operational **health.**

A diagram of a cloud watch

Description automatically generated

CloudWatch observability

AWS X-Ray (**traces**) > Amazon CloudWatch (**metrics, logs**)

* **AWS CloudFormation:**

Create and manage **resources** with **templates**

IaaC

Scale your infrastructure

Manage infrastructure with **DevOps**: Automate, test, and deploy infrastructure templates with continuous integration and delivery (**CI/CD**) **automations**, **Rollback** Support.

Use **helper scripts** for software installation and application setup.

* **AWS CloudTrail:**

Track user activity and API usage

Track API calls

service that enables **governance**, compliance, operational **auditing**, and auditing of your AWS account.

CloudTrail logs, CloudTrail Insights

* **AWS AppConfig:**

Managing application configurations for large-scale configuration.

Assess, audit, and evaluate **configurations** of your **resources**

Track resources inventory and changes

* **Networking & Content Delivery**
* **Amazon Virtual Private Cloud - VPC:**

Isolated cloud resources

Define and launch AWS resources in a logically isolated virtual network

Customize your virtual network by choosing your own **IP address range**, creating **subnets**, and configuring **route tables**.

VPC -> EC2 + RDS, VPCs across accounts, Availability Zones, or AWS Regions

* **Amazon CloudFront:**

Global Content Delivery Network (CDN) with HTTPS communication

Distribution of **static content of web service**

* **Amazon Route 53:**

Scalable domain name system (DNS)

Route end users to Internet apps

provides highly available and scalable [Domain Name System (DNS)](https://aws.amazon.com/route53/what-is-dns/), [domain name registration](https://aws.amazon.com/blogs/networking-and-content-delivery/benefits-of-domain-registration-with-amazon-route-53/), and [health-checking](https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/dns-failover.html) cloud services

* **AWS VPN:**

Securely access your network resources

Connect your on-premises networks and remote workers to the cloud

* **Elastic Load Balancing (ELB):**

Distribute incoming traffic across multiple targets

Security: Secure your applications with SSL/TLS termination, integrated certificate management, and client certificate authentication.

**Automatic scaling**

* **Security, Identity, & Compliance**
* **AWS Identity and Access Management (IAM):**

Securely manage access to services and resources

Grant **temporary security credentials** for workloads that access your AWS resources

IAM policies: east-privilege policies

* **Amazon Cognito:**

Management service for authentication and authorization

Customized sign-up and sign-in

Identity management for your apps

Leverages **IAM roles** to generate **temporary credentials** for your application’s users

**Amazon Cognito user pools**: for external identity providers

* **AWS Secrets Manager:**

Rotate, manage, and retrieve secrets (database credentials, API keys, tokens)

increase security, use this instead of **environment variables** to **store database credentials** and other sensitive information, retrieve them at runtime **without** using **long-term credentials**.

Replicate secrets to support disaster recovery scenarios.

Use AWS Identity and Access Management (**IAM**) permissions **policies** to manage access to your **secrets**.

Integrate secrets with AWS logging, monitoring, and notification services.

* **Environment variables:**

To store secrets securely and adjust your function’s behavior without updating the code.

* **presigned URLs:**

to grant **time-limited access** to objects in Amazon S3 without updating your bucket policy

* **Storage**
* **Amazon Simple Storage Service - S3:**

object-based storage: **key-value** pairs rather than in a file or block structure

Stores data as objects within **buckets**

**Object**: file and metadata that describes the file

**Bucket**: container for objects

**doesn’t** support **HTTPS** for website endpoints.

SQL queries

bucket: logs

Uses:

Data lakes: centralized repository to store structured and unstructured data

backup (databases)

archiving (files)

Content Delivery (images, videos, documents)

static websites (HTML, CSS, JavaScript)

A screenshot of a computer

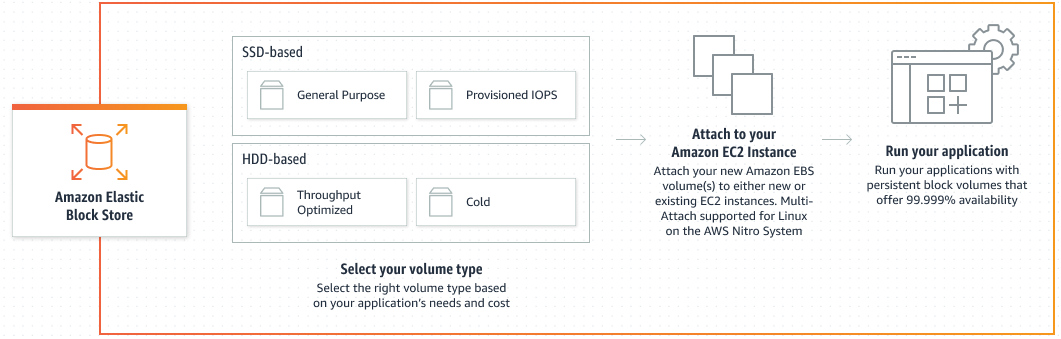
Description automatically generated

* **Amazon Elastic Block Store (EBS):**

**EC2** **block storage** volumes

Uses:

Run relational or NoSQL databases



* **Amazon Elastic File System (EFS):**

Fully managed file system for **EC2**

A screenshot of a computer

Description automatically generated

Uses:

Simplify DevOps

Accelerate data science: for machine learning (ML) and big data analytics workloads

Enhance Content Management Systems (CMS)

* **Others**
* **Amazon Inspector:**

Automated security assessment service that helps improve the security and compliance of applications deployed on AWS.

* **Amazon RDS Proxy:**

Manage database **connections** in serverless environments (AWS Lambda) using connection pool reduces the number of direct connections to the database.

* **Dead Letter Queue (DLQ):**

Configure this on **AWS Lambda** to give you more control over message handling for all **asynchronous** **invocations**, including those delivered via **AWS events** (**S3**, **SNS**, **IoT**). Saves **discarded events** for further processing.

* **AWS Serverless Application Model (AWS SAM):**

<https://aws.amazon.com/serverless/sam/>

simplifies the development and deployment of serverless applications.

Develop, debug, and deploy your serverless applications with the **AWS SAM CLI**.

Define and manage your infrastructure code with **AWS SAM templates**.

Perform **real-time debugging** and **testing** in the cloud with **AWS SAM Accelerate**.

consists of two parts:

AWS SAM **templates**: IaC

AWS Serverless Application Model Command Line Interface (**AWS SAM CLI**): create, develop, and deploy serverless applications

**sam build** and **sam deploy** commands to prepare your application for deployment and deploy to AWS.

**sam sync** command to watch for local changes and quickly deploy those changes to AWS. Then, use **sam remote invoke** to test your lambda functions in the cloud.

**sam pipeline** command to create or modify pipelines for your continuous integration and continuous delivery (CI/CD) system.

**SAML directory:** You cannot provide access to an on-premises SAML directory using a VPC endpoint.

