



AWS Certified Developer – Associate

Richard Spencer

Objectives:

Demonstrate an understanding of core AWS services, uses, and basic AWS architecture best practices. Demonstrate proficiency in developing, deploying, and **debugging** cloud-based applications using AWS.



AWS Certified Developer - Associate Exam

AWS Certified Developer - Associate (DVA-C01)

- Intended for individuals who perform a development role and have one or more years of hands-on experience developing and maintaining an AWS-based application.
- Skills validated by this certification
 - Demonstrate an understanding of core AWS services, uses, and basic AWS architecture best practices
 - Demonstrate proficiency in developing, deploying, and debugging cloud-based applications using AWS
- Intended for individuals who perform a development role and have one or more years of hands-on experience developing and maintaining an AWS-based application.
- Format:
 - Multiple Choice (Has one correct response and three incorrect responses (distractors))
 - Multiple Answer (Has two or more correct responses out of five or more options)



AWS Certified Developer - Associate Exam

AWS Certified Developer - Associate (DVA-C01)

- Time: 130 minutes
- Scaled score between 100 and 1,000
- Passing score is 720
- Price: \$150 USD



Content Domains

Domain	% of Scored Items
Section 1.0: Deployment	22%
Section 2.0: Security	26%
Section 3.0: Developing with AWS Services	30%
Section 4: Refactoring	10%
Section 5: Monitoring and Troubleshooting	12%
TOTAL:	100%

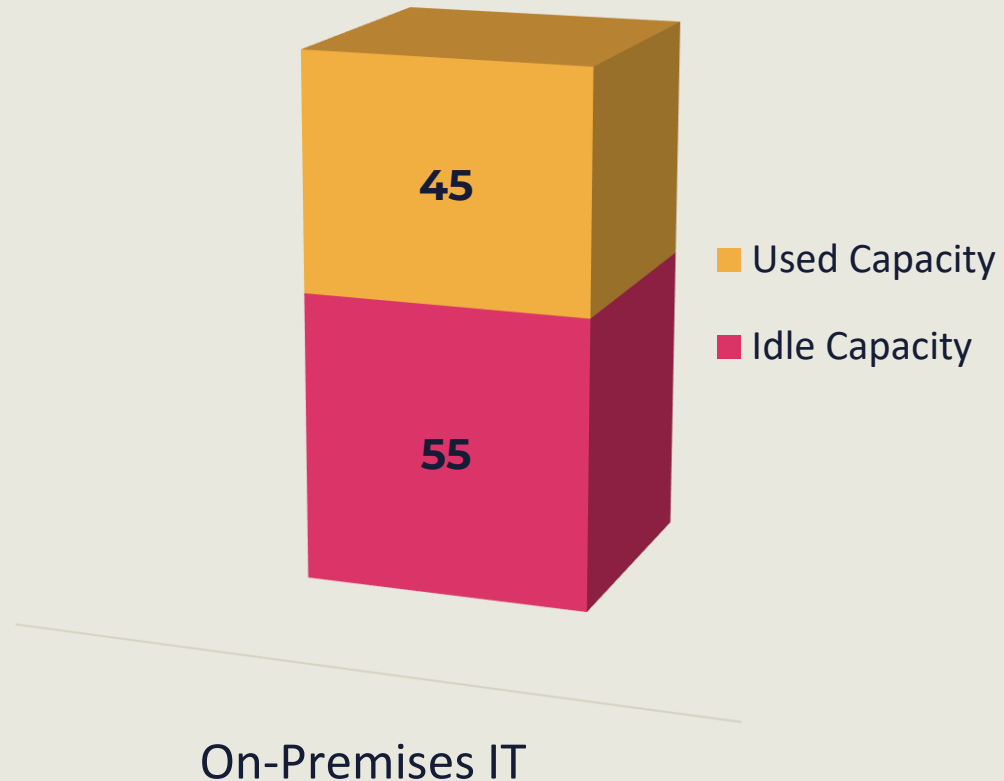


Recommended Knowledge and Experience

- In-Depth knowledge of at least one high-level programming language
- Understanding of core AWS services, uses, and basic best practices
- Ability to use the AWS service APIs, AWS CLI, and SDK's to write applications
- Understanding of the AWS shared responsibility model
- Understanding of application lifecycle management
- Ability to use a CI/CD pipeline to deploy applications on AWS
- Ability to write code using AWS security best practices
- Ability to author, maintain, and debug code modules on AWS
- Proficiency writing code for serverless applications
- Understanding the user of containers in the development process

What is Total Cost of Ownership (TCO) ?

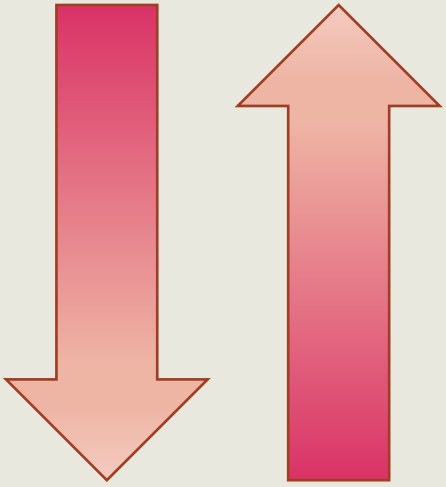
- Used to compare total cost of ownership (acquisition and operational costs) for running an infrastructure environment on premises, versus in the AWS cloud.
- A study from 2017 states that typical data centers are 45% utilized
- Paying for AC, Power, Maintenance, even when servers are not in use



Comparing TCO is complex

Server Costs	Hardware – Server, Rack, Chassis, Maintenance	Software – OS, Virtualization Licenses	Facilities Cost		
			Space	Power	Cooling
Storage Costs	Hardware – Storage Disks, NAS switches	Software - Backup	Facilities Cost		
			Space	Power	Cooling
Network Costs	Network Hardware – LAN Switches, Routers, Bandwidth	Software – Network Monitoring	Facilities Cost		
			Space	Power	Cooling
IT Labor Costs	Server Admin, Virtualization Admin, Storage Admin, Network Admin, Support Team				
Extras	Project Planning, Advisors, Legal, Contractors, Managed Services, Training, Capital Costs				
Business Value: Cost of delays Risk premium Competitive abilities Governance					

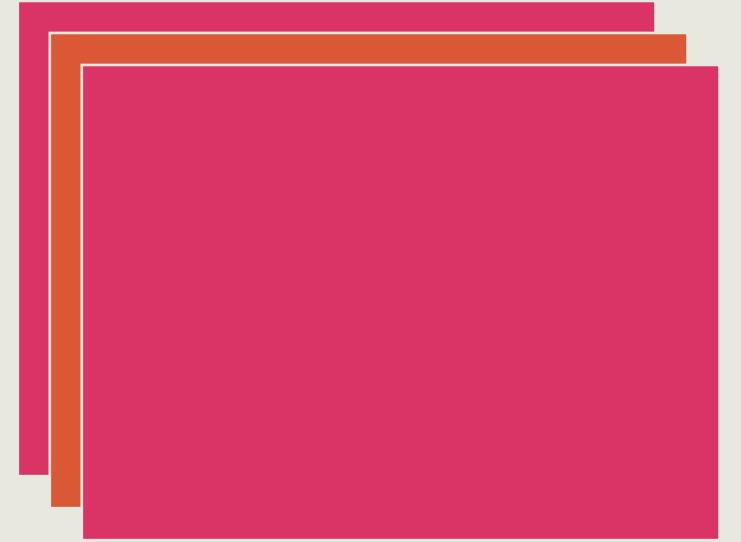
How do we calculate our TCO?



Match appropriate supply
with appropriate demand
(Right Sizing)



Matching supply and demand
with the use of elasticity



Lowering unit prices through
reserved or spot instances

On-Premises vs AWS

On-Premises

Purchase cost + annual
maintenance fee

Purchase software and hardware
+ annual maintenance fee

Purchase + annual maintenance
fee

OS + Virtualization software –
licensing + support

Application licensing and support

Manpower + managed services

AWS

EC2

EBS/EFS/S3/Glacier

Direct Connect + Data transfer

Purchase cost + annual
maintenance fee

Application licensing and support

Manpower + managed services

Server / Compute

Storage

Networking

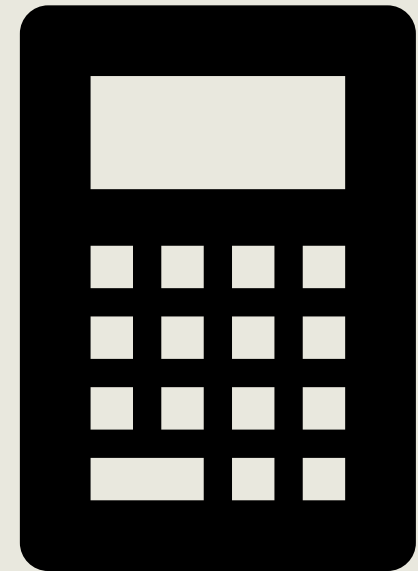
Software

Application

Management

Introducing AWS Pricing Calculator

- Preceded by AWS TOC (Total Cost of Ownership) Calculator
- The AWS Pricing Calculator provides an estimated cost per month.
- Create Estimate, select the resources you intend to use, and an estimated monthly bill will be provided
- Accuracy of calculator is dependent on the accuracy of the information you supply
- Not to be confused with AWS Cost Explorer which lets you visualize and manage your costs and usage over a period of time



Introduction to IAM

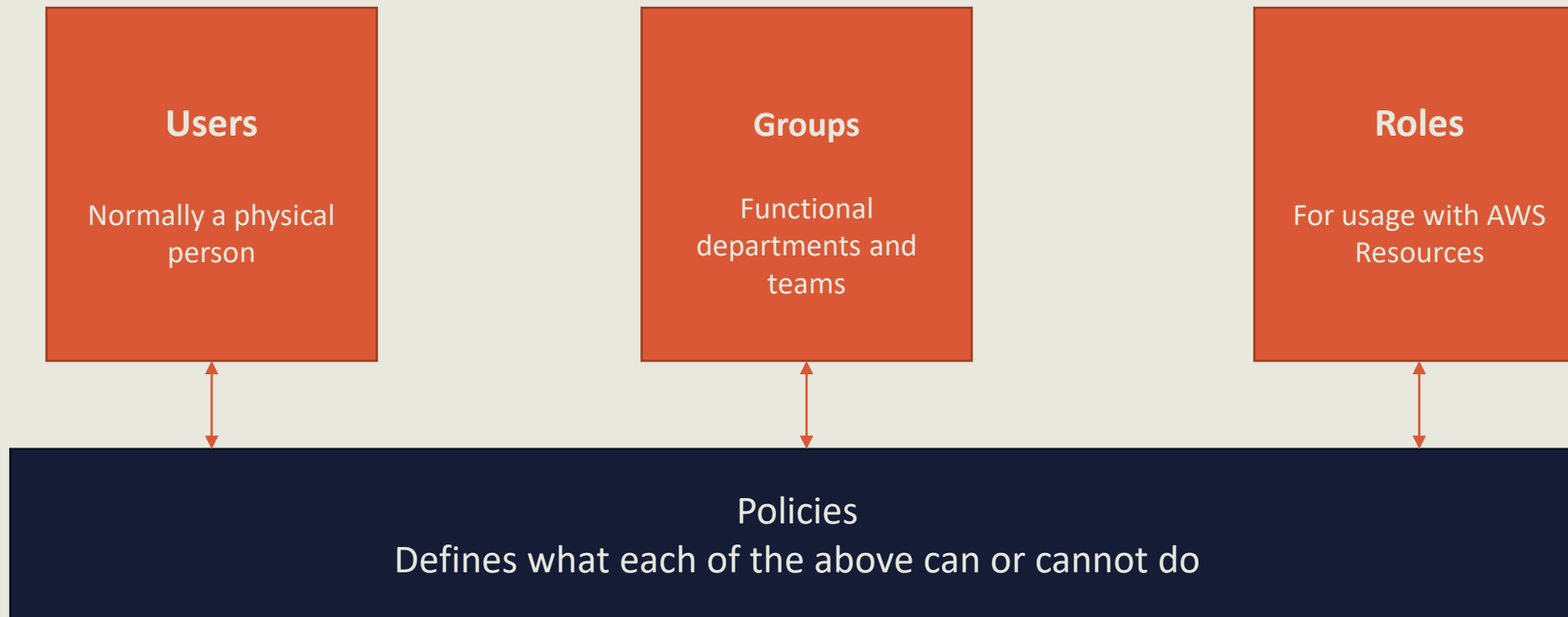
- IAM (Identity and Access Management)
 - Security center of your cloud
- Manage IAM users and their access
 - User role vs Service role
- Best practices for securing your AWS account through IAM
- Manage access for federated users
- AWS Security Token Service (STS)
- Amazon Cognito



Introduction to IAM

- IAM (Identity and Access Management)
 - All AWS security is controlled here, including:
 - Users
 - Groups
 - Roles
 - Grant different permissions to different people for different resources
 - Make use of an access key ID and secret access key to make programmatic (CLI, SDK) requests to AWS
 - You have a unique account sign-in page:
https://my_aws_account_id.signin.aws.amazon.com/console
 - Policies are written in JSON (JavaScript Object Notation)

Introduction to IAM



Introduction to IAM

- Identity-based policies
 - Essential to provide users with permissions to access AWS resources in their own account
 - Attached to an IAM user, group, or role
 - Control what they can and cannot do
- Resource-based policies
 - Attached to a resource such as S3 buckets, SQS queues, and AWS KMS
 - Used to specify who has access to a resource and what actions they can perform on it
 - Inline only, not managed



Introduction to IAM

- Best Practices
 - Do not use or share the AWS Account Root user access keys
 - Create individual IAM users
 - Use Groups to assign permissions to IAM users
 - Grant least privilege
 - Strong Password Policy
 - Do not share keys
 - Rotate credentials regular / delete unnecessary credentials
 - Monitor Activity in your AWS Account (CloudTrail)



Introduction to IAM

- Federated Users
 - If your organization is already using an authentication mechanism, you can federate these users into your AWS account
 - Enables a separate SAML 2.0 or Open ID Connect (OIDC) IdP
 - Pass attributes from local identity provider, to AWS such as title, team, and job role
 - Implement access permissions based on attributes passed in





Introduction to IAM

- Amazon Short Token Service (STS)
 - Enables you to create and provide users with temporary security credentials
 - Short lived credentials and not stored with user
 - Generated dynamically and provided when user requests it
 - Pass attributes from local identity provider, to AWS such as title, team, and job role
 - Global Service: <https://sts.amazonaws.com>



Introduction to IAM

- AWS STS Role Options
 - AssumeRole
 - Will give you a set of temporary security credentials that you can use to get access to some of the AWS resources that you normally wouldn't have access to
 - Consists of access key ID, secret access key, and security token
 - Used within your own account or cross-account access
 - AssumeRoleWithSAML
 - Will give you a set of temporary security credentials for users that have been authenticated via a SAML authentication response
 - AssumeRoleWithWebIdentity
 - Will give you a set of temporary security credentials for users that have been authenticated in a mobile or web application with a web identity provider

Introduction to IAM

- AWS STS Use Cases
 - Good for assuming roles in another account
 - Enforces MFA for CLI
 - Apply permissions to user in another identity store (ie. Active Directory) using services like Cognito



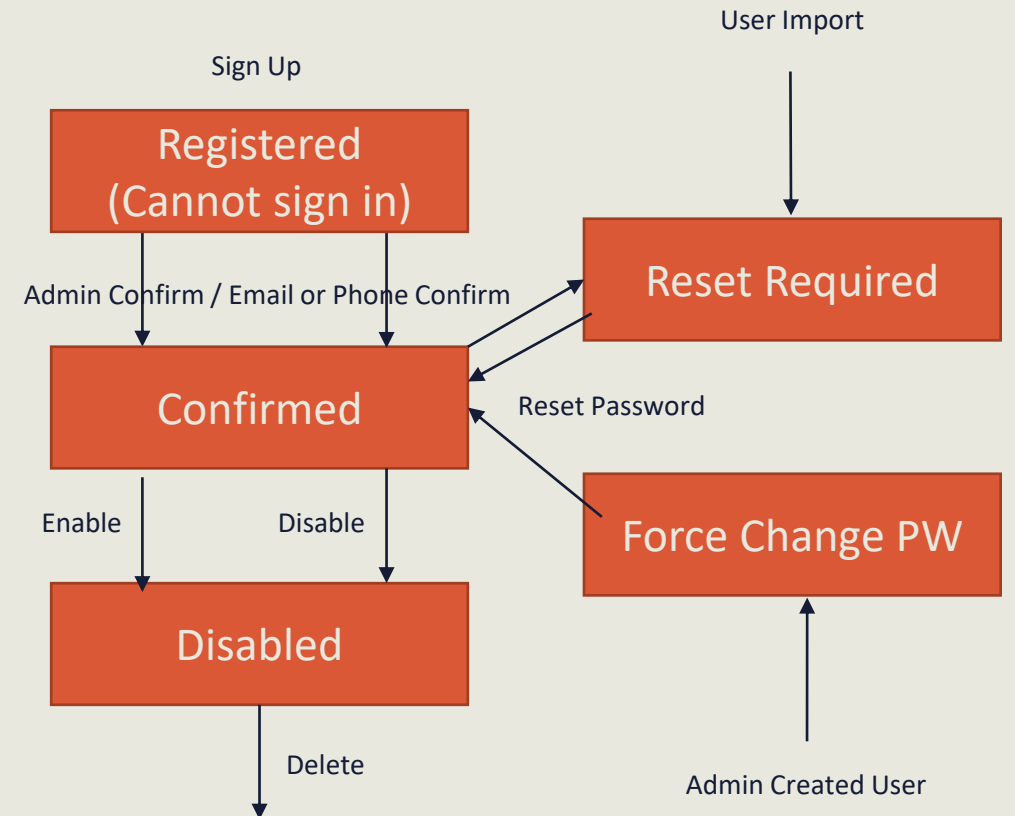


Introduction to IAM

- Amazon Cognito
 - Used for mobile and web authentication
 - Usually grouped with Lambda, API Gateway, and DynamoDB with regards to exam questions
 - Developer – Associate exam will ask questions along the lines of , what can we use to make an sign up/sign in feature for our mobile applications.
 - Authenticate users through external Identity Provider and provide temporary security credentials
 - Works with SAML or OpenID connect, and social identity providers such as Facebook, and Twitter)
 - Can also integrate your own identity provider
 - Great for adding user management and sync functionality to your web apps

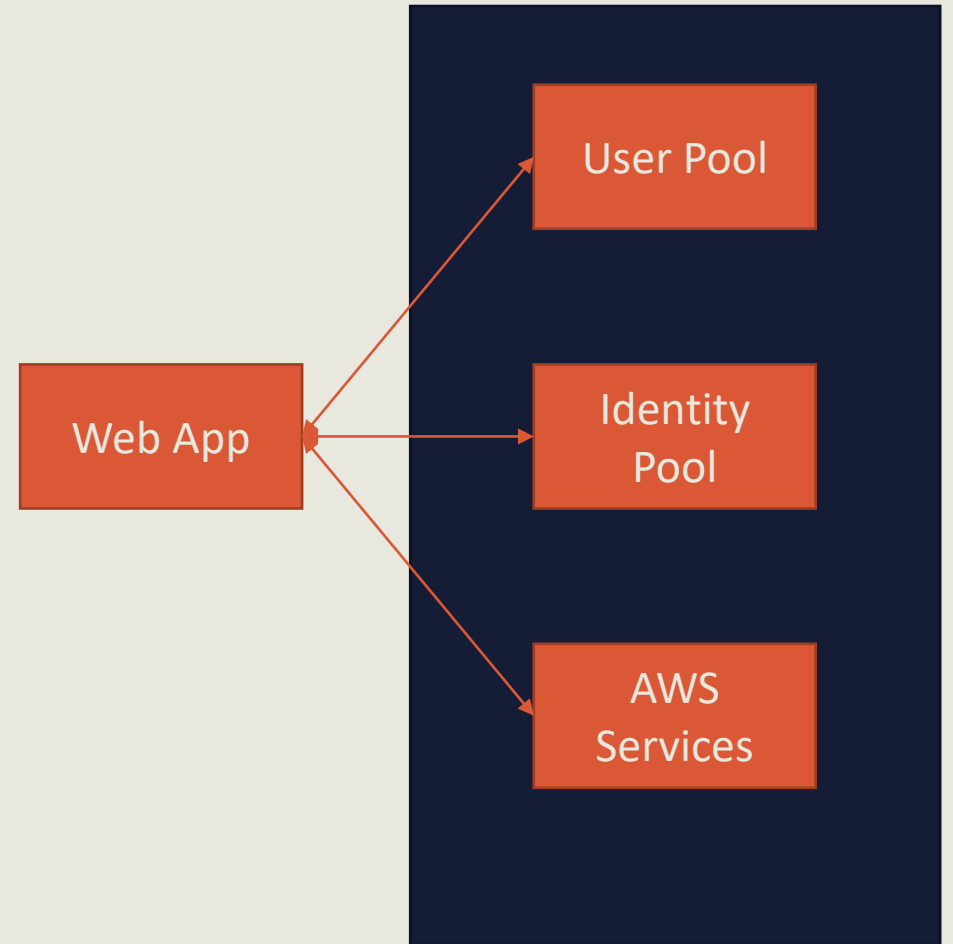
Introduction to IAM

- Amazon Cognito User Pools
 - Directories created for sign-up/sign-in options for app users
 - Can sign in through Cognito or 3rd party IdP
 - User Pools are region specific



Introduction to IAM

- Amazon Cognito Identity Pools
 - Used to federate your users to your AWS Services
 - Grant your users temporary AWS credentials
 - Supports identity providers such as:
 - Amazon Cognito user pools
 - Social sign-in with Facebook, Google, and Amazon
 - OpenID Connect and SAML identity providers
 - Developer authenticated identities
 - Permissions for each user are controlled through IAM roles



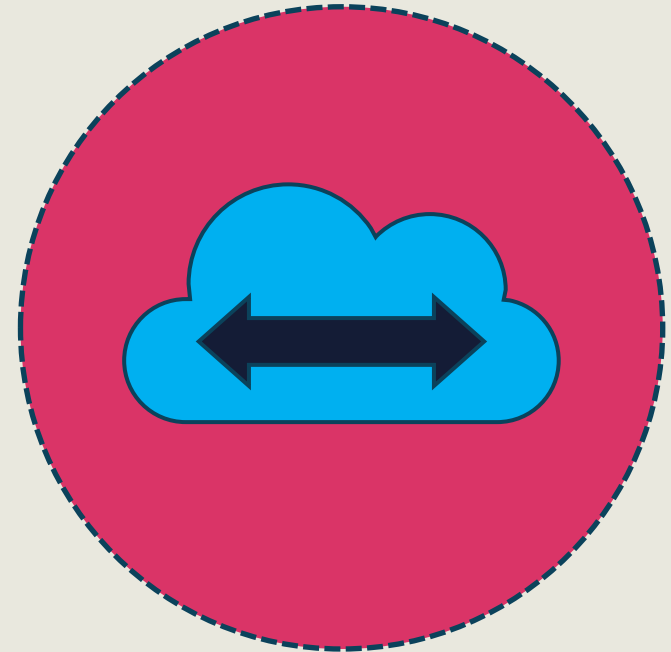


Introduction to IAM

- IAM – AWS Developer Associate Exam Tips
 - Create only 1 IAM User per PERSON
 - Create only 1 IAM Role per Application
 - Credentials should always be kept private
 - Never code your credentials into your application
 - Retire ROOT account after initial setup
 - Enable MFA

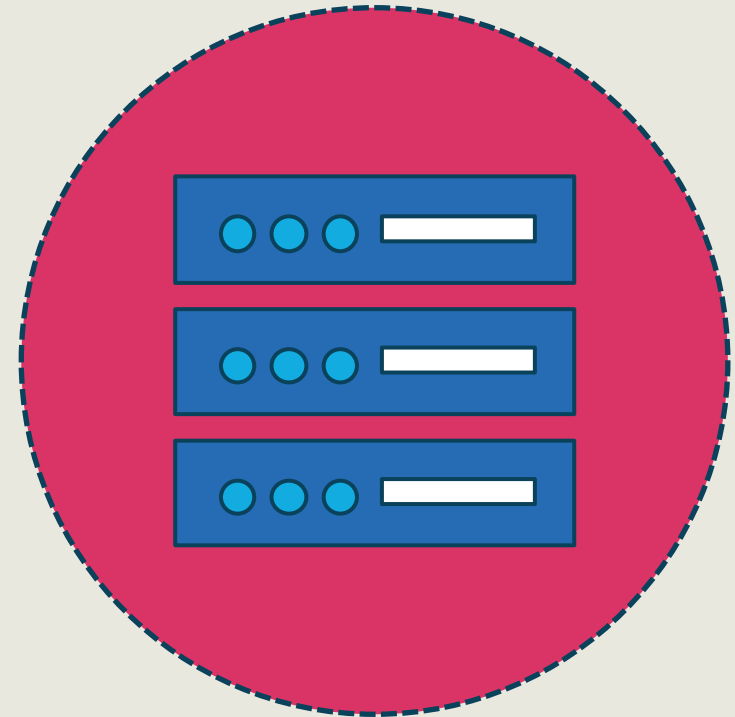
Amazon Compute Services

- In this section we will learn AWS Compute concepts, including:
 - Elastic Compute Cloud (EC2)
 - Types and families
 - Auto Scaling
 - Ephemeral vs persistent storage
 - Amazon Machine Images (AMIs)
 - Bootstrapping/user data

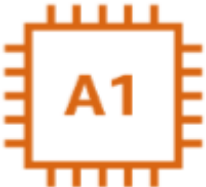
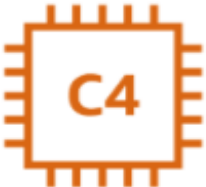




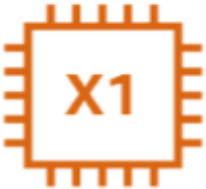




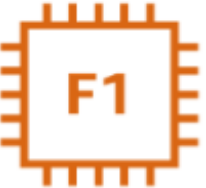



Amazon Compute Services

- Amazon EC2
 - Designed to make web-scale computing easier for developers
 - Choose Linux or Windows
 - SSH or Remote Desktop functionality
 - Deploy across AWS Regions and Availability Zones
 - **Resizable** compute capacity
 - **Complete** control of your computing resources
 - **Reduced** time required to obtain and boot server instances
 - Pay for only the compute capacity that you use
 - Tools to build failure-resilient applications



Amazon Compute Services

General Purpose	Compute Optimised	Memory Optimised	Accelerated Computing	Storage Optimised
 ARM based core and custom silicon	 Compute - CPU intensive apps and DBs	 RAM - Memory intensive apps and DB's	 Processing optimised- Machine Learning	 High Disk Throughput - Big data clusters
 Tiny - Web servers and small DBs		 Xtreme RAM - For SAP/Spark	 Graphics Intensive - Video and streaming	 IOPS - NoSQL DBs
 Main - App servers and general purpose		 High Compute and High Memory - Gaming	 Field Programmable - Hardware acceleration	 Dense Storage - Data Warehousing

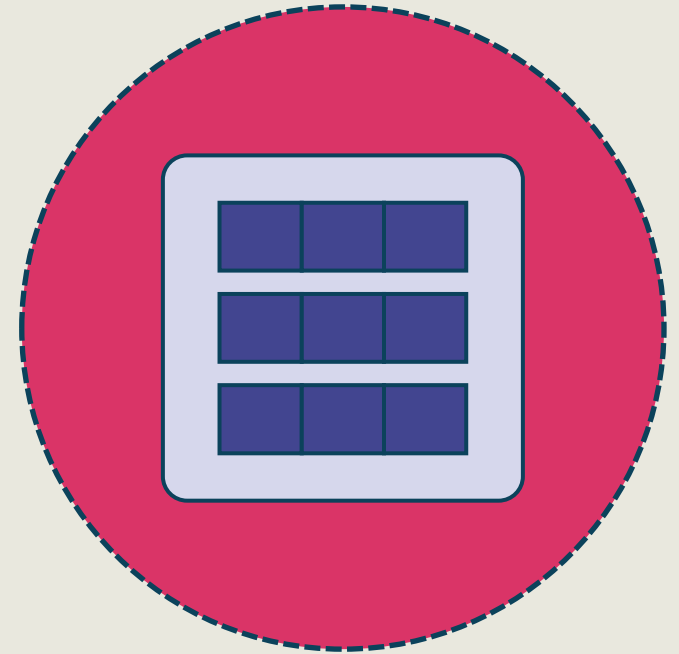
Amazon Compute Services

- Amazon EC2 Storage
 - Instance Store
 - Ephemeral
 - Only to be used for temporary storage
 - Does not persist after shutdowns, terminations, or failures
 - Elastic Block Store (EBS)
 - Persistent
 - Backup via snapshots
 - Encryption supported
 - Can be moved across instances



Amazon Compute Services

- Amazon EC2 and AMIs
 - An AMI is a template that contains a software configuration that includes operating systems, application servers, and applications
 - An AMI includes:
 - A template for the **root volume** for the instance
 - **Launch permissions** that control which AWS accounts can use the AMI to launch instances
 - A block device mapping that specifies the volumes to attach to the instance when it is launched



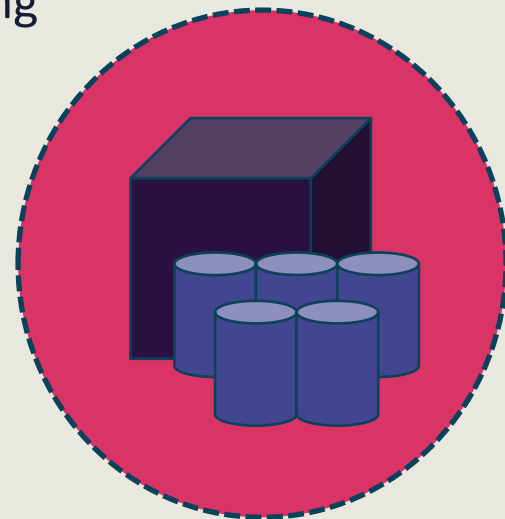


Amazon Compute Services

- Amazon EC2 instance User Data, and instance metadata
 - User Data
 - Passed to the instance at launch
 - Perform common automated configuration tasks
 - Runs scripts after instance starts
 - Can be Linux script or Windows batch / Powershell scripts
 - Run once per instance ID
 - Metadata
 - Data about your instance
 - Used to configure or manage a running instance
 - <http://169.254.169.254/latest/meta-data/>

Amazon Storage Services

- Amazon S3
 - Storage for the internet
 - HTTP access
 - Store and retrieve any amount of data, any time, from anywhere on the web
 - Infinitely scalable, reliable, fast and durable
 - Great for media hosting, snapshots, application file hosting, static website hosting
 - Stores data as objects within buckets
 - Object is composed of a file and any metadata with the file
 - Full control over who has access to the bucket and its objects
 - Versioning and lifecycle management





Amazon Storage Services

- Amazon Elastic Block Store (EBS)
 - Persistent block level storage volumes
 - Data is automatically replicated within its availability zone
 - Snapshots stored in Amazon S3
 - Used with EC2 to provide persistent storage
 - Choice of SSD or HDD
 - SSD for transactional workloads with frequent read/write operations and little I/O
 - HDD for large streaming workloads with high throughput

Amazon Storage Services

- Amazon Elastic File System (EFS)
 - Easy to use, scalable, NFS file system to use with AWS services or on-prem resources
 - Scales into the petabytes
 - Grows and shrinks automatically
 - Used to share access to thousands of EC2 instances
 - Good for aggregating throughput and IOPS
 - Also good for shared directories

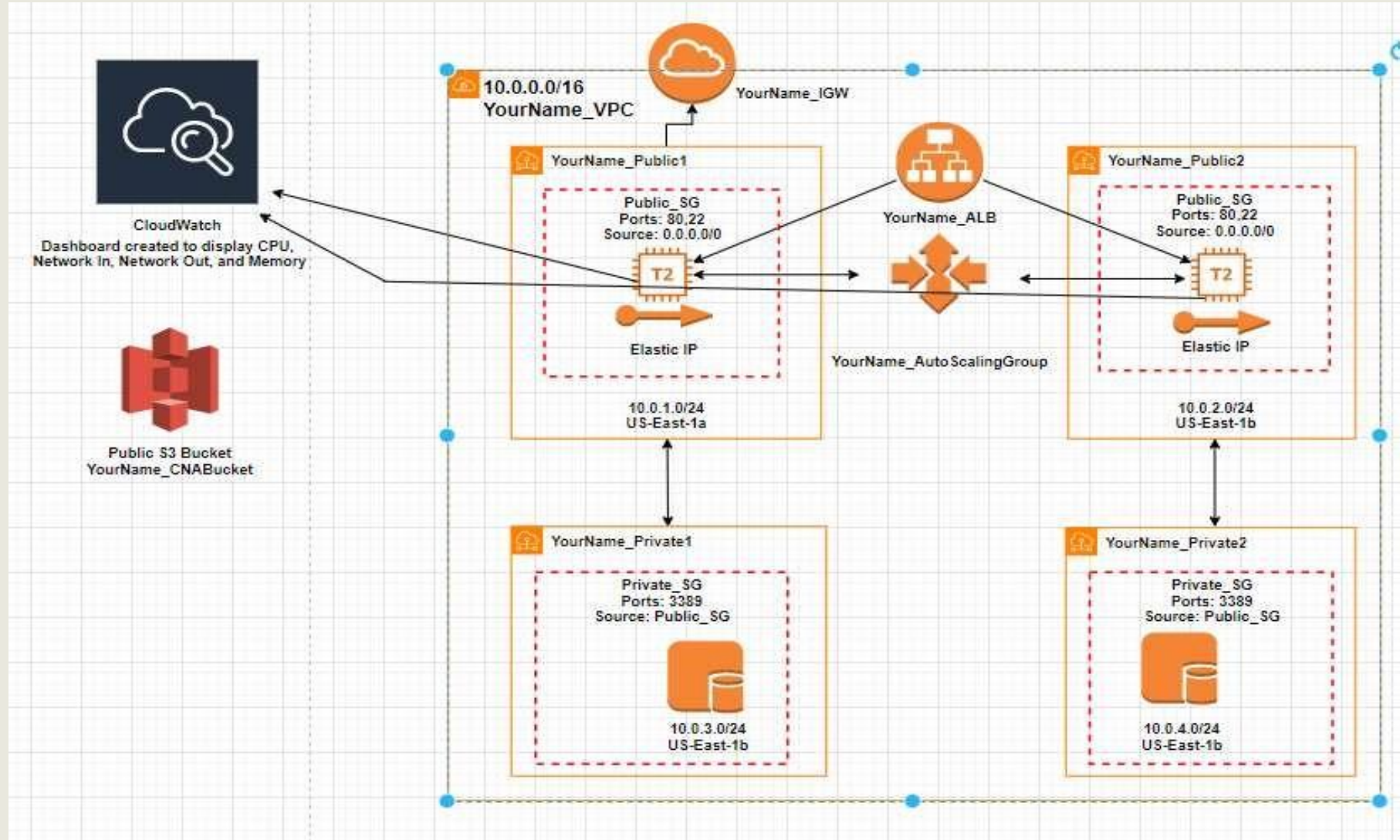








Amazon Network Services

- Amazon Virtual Private Cloud (VPC)
 - Private, isolated virtual network in the cloud
 - Create multiple subnets over different availability zones
 - Public vs Private subnets (NAT Gateway)
 - Secured via:
 - Security Groups
 - Network ACLs
 - Key Pairs
- VPN connectivity to on-premises
 - Hardware/Software VPN
 - Direct Connect

Amazon Network Services

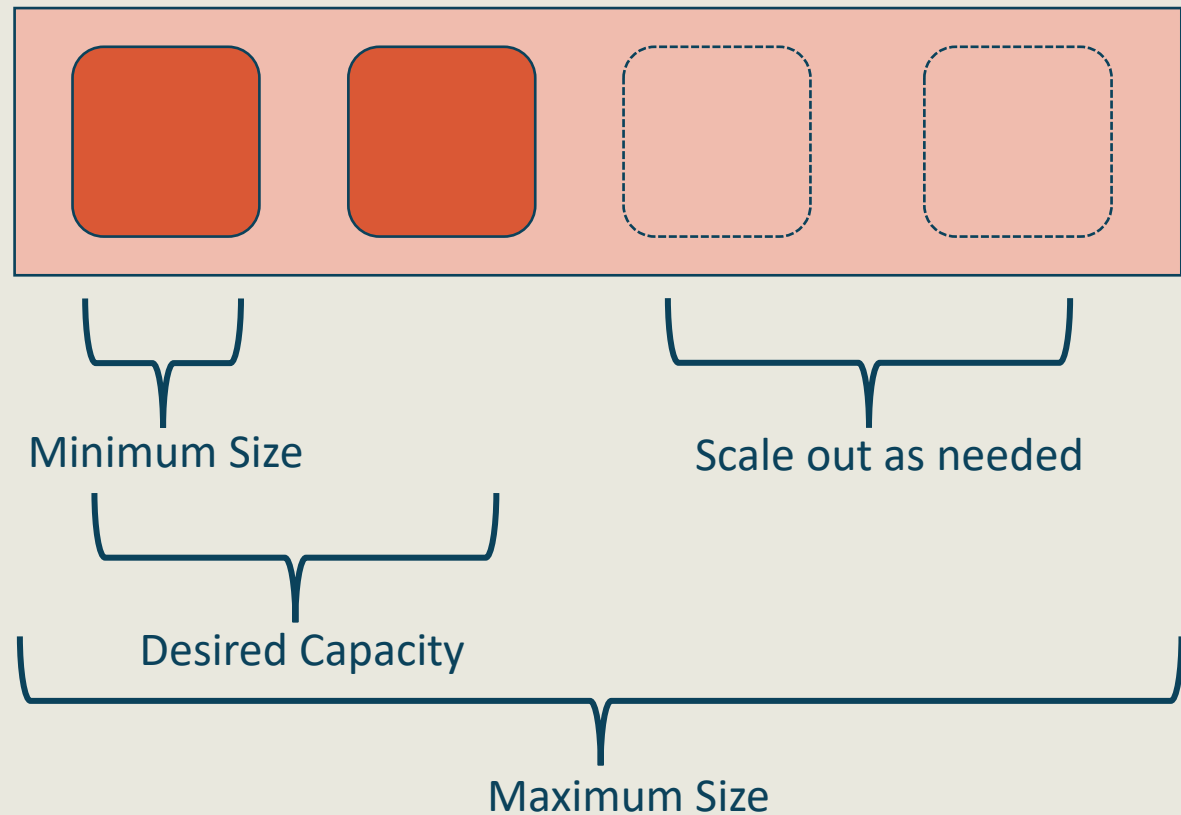


Amazon Database Services

 RDS	 DynamoDB	 Redshift	 ElastiCache
Fully managed Relational Database Service	Fully managed NoSQL database service	Fully managed data warehouse service	In-memory data store and cache
Highly available (Multi-AZ)	Highly available	Highly available	Extremely fast performance
Highly scalable	Store any amount of data with no limits	Peta-byte scale	Highly scalable
Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle, Microsoft SQL Server	Provides fast performance using SSD	Create clusters to upload data to	Popular for caching, session stores, gaming leaderboards, real-time analytics, and queuing
Easily migrate existing database to Managed RDS via Database Migration Service	Can easily provision and change request capacity needed for each table	Can perform data analysis queries	Two engines: ElastiCache for Redis ElastiCache for Memcached

Amazon Elasticity and Management Tools

- Amazon EC2 Autoscaling
 - Enhanced fault-tolerance
 - Improved availability
 - Easier to manage costs



Amazon Elasticity and Management Tools

Classic Load Balancer	Application Load Balancer	Network Load Balancer
EC2-Classic	Path-based routing	High throughput
VPC	Containerized applications	Low latency
TCP and SSL listeners	Monitoring health of each service independently	Static IP address support
Sticky sessions	Route requests to multiple services on a single EC2 instance	Designed to handle tens of millions of requests per second
OSI Layer 4 (Network Layer)	OSI layer 7 (Application Layer)	OSI Layer 4 (Network Layer)

Amazon Elasticity and Management Tools

- Monitoring and Management Tools
 - Amazon Cloudwatch
 - Visible info on resource utilization, and other metrics
 - Custom metrics available
 - Accessible in Console, APIs, SDK, and CLI
 - Amazon CloudTrail
 - Track all actions and API calls in your account
 - Useful for governance, compliance, and operational auditing
 - Amazon X-Ray
 - Useful in debugging and analyzing application issues,

Interacting with Amazon Web Services

Amazon Web Services Console

Web based, Graphical User Interface

Amazon Web Services CLI

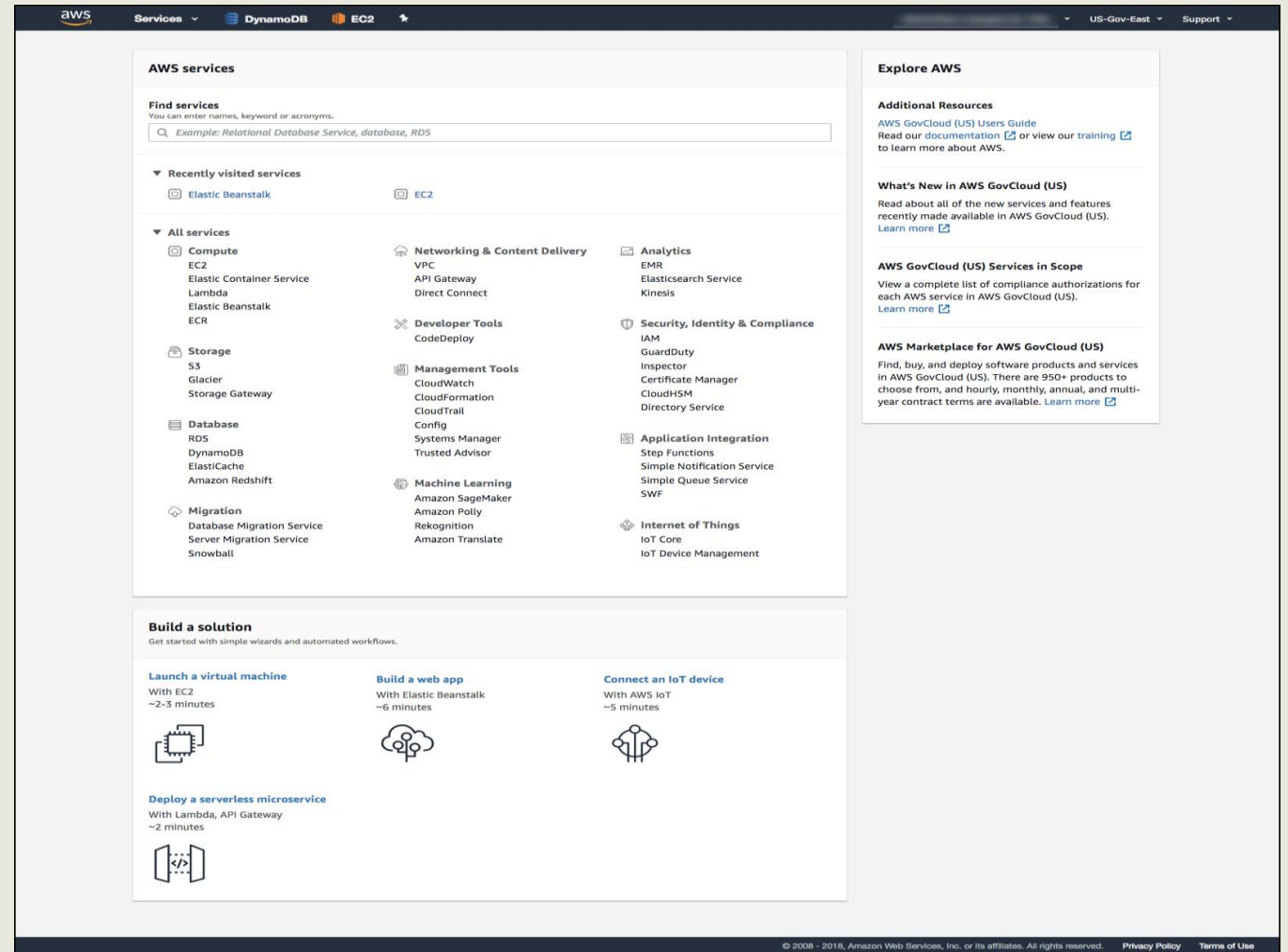
Windows and Linux Command Line Interface

Amazon Web Services SDKs

Software Development Kits in many languages

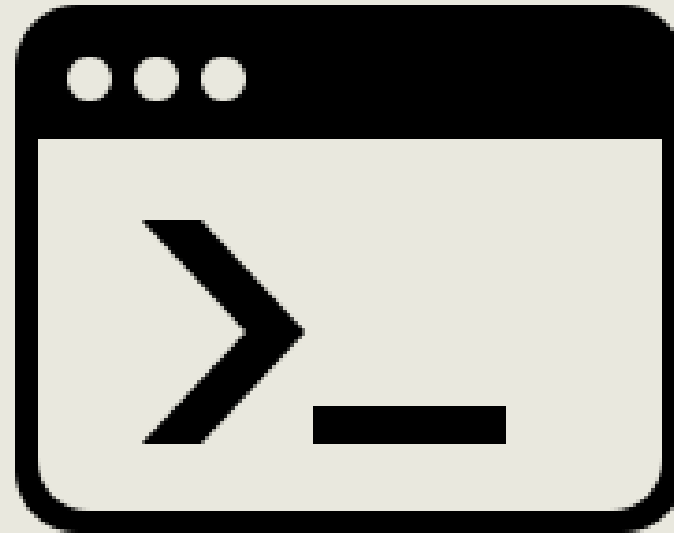
Interacting with Amazon Web Services

- AWS Console
 - Web application with user interface
 - Perform tasks on all AWS services
 - Each service has its “dashboard”
 - Can rearrange console for easy access
 - Customized sign-in URL
 - IAM users



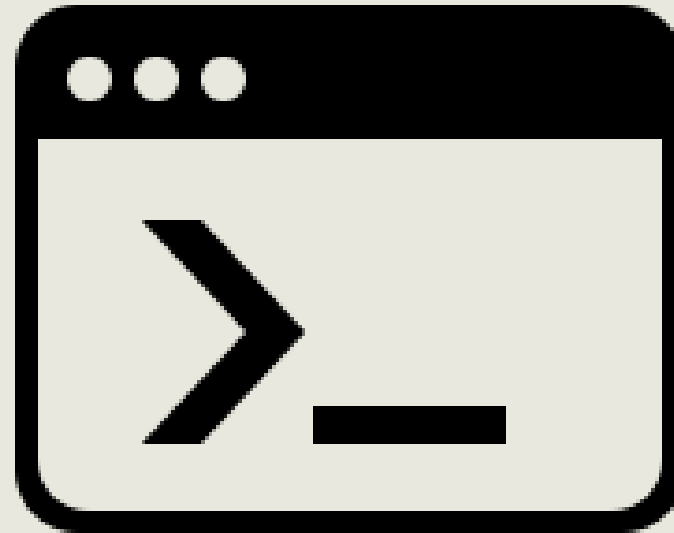
Interacting with Amazon Web Services

- AWS CLI
 - Linux or Windows Command Line Interface
 - Great for automating processes via scripts
 - Easier to load large files to S3
 - Supports all Amazon services
 - Faster, from a development perspective (no logging in, etc)
 - Built-in to many AMIs



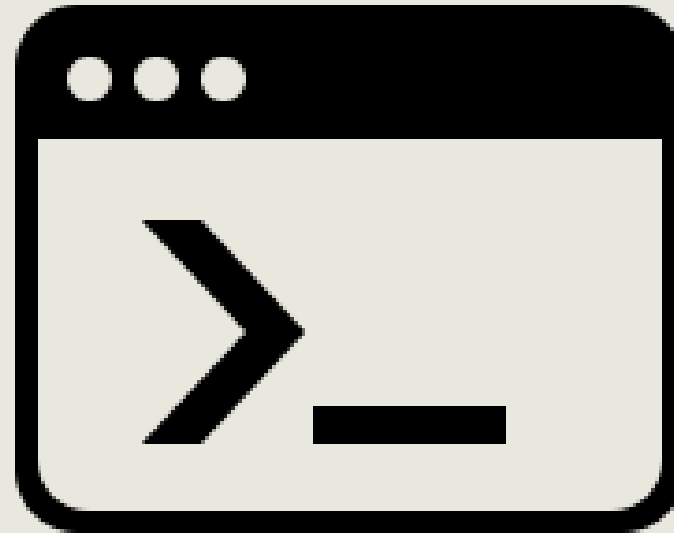
Interacting with Amazon Web Services

- Installing the AWS CLI
 - AWS CLI can be installed in three ways:
 - Using pip
 - Using a virtual environment
 - Install into isolated Python environment
 - Using a bundled installer
 - CLI version 2 only



Interacting with Amazon Web Services

- Installing via pip
 - Pip is a package management program used to install and manage software written in Python
 - Prerequisites:
 - Python 2 v 2.6.5+ or Python 3 v 3.3+
 - Windows, Linux, MacOS, or Unix OS
 - Two versions:
 - Version 1: Still supported
 - Version 2: Self contained, embedded copy of Python included in the installer



Interacting with Amazon Web Services

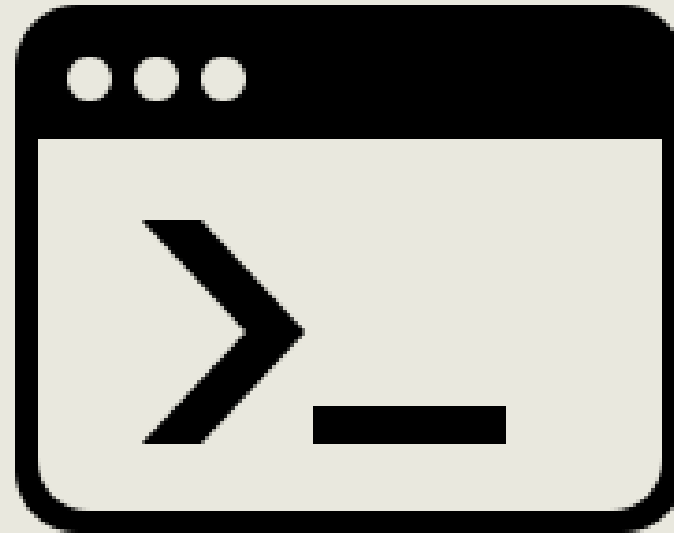
- Configuring the AWS CLI
 - You will need:
 - Your Access key ID
 - Your Secret Access Key
 - Default Format (json)
 - Region

```
cmd Select Command Prompt
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Richard>aws configure
AWS Access Key ID [*****O2P3]:
AWS Secret Access Key [*****sTS7]:
Default region name [us-east-1]:
Default output format [json]:
```

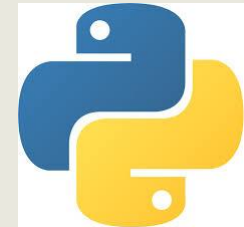
Interacting with Amazon Web Services

- Common CLI Commands
 - To launch an EC2 instance
 - `aws ec2 run-instances --image-id ami-xxxxxxx --count 1 --instance-type t2.micro --key-name MyKeyPair --security-group-ids sg-903004f8 --subnet-id subnet-6e7f829e`
 - To list S3 buckets
 - `aws s3 ls`
 - To see host inventory
 - `aws ec2 describe-hosts --filter "Name=state,Values=available"`



Interacting with Amazon Web Services

- AWS SDKs
 - A collection of software tools for the creation of applications and libraries that use Amazon Web Services (AWS) resources
 - Develop applications on AWS in the programming language of your choice



Interacting with Amazon Web Services

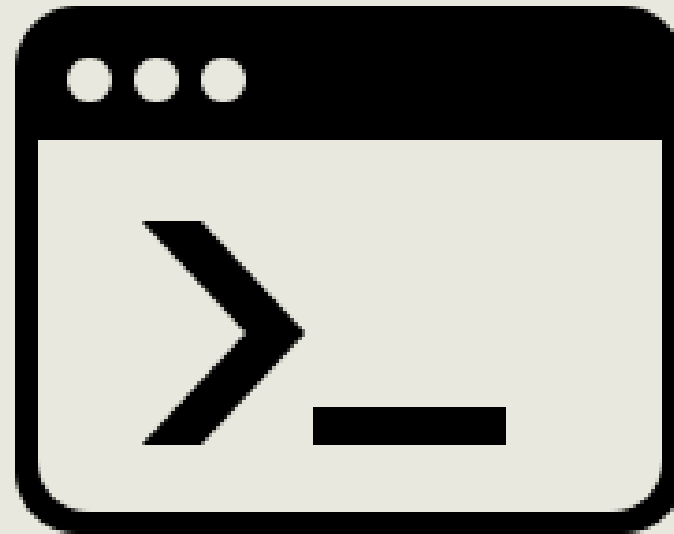
- AWS SDKs
 - We can also browse SDKs based on what we want to use them for. We can choose from:
 - IoT Device SDKs
 - Arduino Yun
 - Embedded C
 - Mobile SDKs
 - Android
 - iOS
 - React Native
 - Mobile Web JS



Android

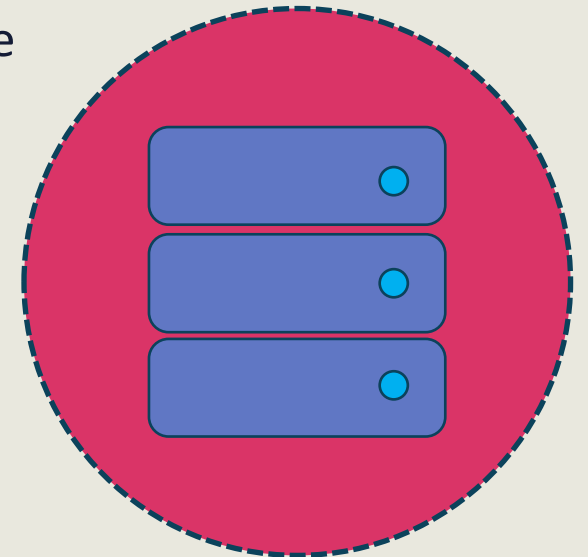
Interacting with Amazon Web Services

- Amazon EC2 API (SDK Alternative)
 - Web service used to launch and manage EC2 instances.
 - Rest based API service that allows you to take action on the following AWS services:
 - Amazon EC2
 - DescribeInstances
 - Amazon EBS
 - EnableEbsEncryptionByDefault
 - AttachVolume
 - Amazon VPC
 - CreateVpc



AWS Storage Services

- EC2 Instance Storage
 - Temporary block-level storage for instances
 - Convenient for ever changing information such as buffers, cache, or scratch data
 - Located on physically disks attached to host computer
 - Lowest latency storage available to instance
 - Ephemeral storage!! Data persists only during lifetime of associated instance
 - Will lose data if drive fails, instance is stopped or terminated
 - HDD and SSD volumes
 - Still provide value with its massive IOPS at low latency
 - Cannot perform resizing





AWS Storage Services

- Elastic Block Storage (EBS)
 - Remote network storage connected via SAN or NAS
 - Persistent, block level storage for use with EC2 instances
 - Great for databases, file systems, and applications that need access to raw, unformatted storage
 - No need for initialization, max performance from the start
 - 5,000 EBS volumes by default
 - Options (up to 16 TiB in size):
 - General Purpose SSD (gp2)
 - Provisioned IOPS SSD (io1)
 - Throughput Optimized HDD (st1)
 - Cold HDD (sc1)



AWS Storage Services

- Elastic Block Storage (EBS)
 - Mountable to multiple instances (EBS Multi-attach)
 - Multiple volumes mountable to one instance
 - Can use encryption
 - Point in time snapshots that can be copied across regions
 - Used to move volumes across Azs as EBS volumes are locked to an AZ
 - Provisioned capacity in GBs and IOPs
 - Billed for all the provisioned capacity

AWS Storage Services

Solid-state drives (SSD)			Hard disk drives (HDD)	
Volume type	General Purpose SSD (gp2)	Provisioned IOPS SSD io2 io1	Throughput Optimized HDD (st1)	Cold HDD (sc1)
Description	General purpose SSD volume that balances price and performance for a wide variety of workloads	Highest-performance SSD volume for mission-critical low-latency or high-throughput workloads	Low-cost HDD volume designed for frequently accessed, throughput-intensive workloads	Lowest cost HDD volume designed for less frequently accessed workloads
Use cases	<ul style="list-style-type: none"> •Recommended for most workloads •System boot volumes •Virtual desktops •Low-latency interactive apps •Development and test environments 	<ul style="list-style-type: none"> •Critical business applications that require sustained IOPS performance, or more than 16,000 IOPS or 250 MiB/s of throughput per volume •Large database workloads, such as: <ul style="list-style-type: none"> •MongoDB •Cassandra •Microsoft SQL Server •MySQL •PostgreSQL •Oracle 	<ul style="list-style-type: none"> •Streaming workloads requiring consistent, fast throughput at a low price •Big data •Data warehouses •Log processing •Cannot be a boot volume 	<ul style="list-style-type: none"> •Throughput-oriented storage for large volumes of data that is infrequently accessed •Scenarios where the lowest storage cost is important •Cannot be a boot volume
Max IOPS per volume	16,000 (16 KiB I/O) *	64,000 (16 KiB I/O) †	500 (1 MiB I/O)	250 (1 MiB I/O)
Max throughput per volume	250 MiB/s *	1,000 MiB/s †	500 MiB/s	250 MiB/s

AWS Storage Services

- Amazon Simple Storage Service (S3)
 - Object level storage within buckets
 - Object = file + any metadata
 - Key is unique ID for an object in a bucket
 - Unlimited storage capacity





AWS Storage Services

- Amazon S3 Buckets
 - Bucket name must be a unique name across all of Amazon S3
 - Names are not able to be changed
 - You can control access via bucket policies and IAM
 - View access logs for the bucket and any objects
 - Choose which geographical region you want to store the bucket (cannot be changed)
 - Great for hosting static websites
 - Data Consistency:
 - Read-after-write for PUTS of new objects
 - Eventual consistency for read-after-write HEAD or GET requests
 - Eventual consistency for overwrite PUTS and DELETES in all regions

AWS Storage Services

- Amazon S3 Bucket Security
 - Bucket Policies
 - Specify what actions are allowed or denied for which principals on the bucket, that the bucket is attached to
 - Ie. Deny user Richard to PUT but not DELETE objects in bucket

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": ["arn:aws:iam::111122223333:user/Richard",
               "arn:aws:iam::111122223333:root"]
      },
      "Action": "s3:*",
      "Resource": ["arn:aws:s3:::my_bucket",
                  "arn:aws:s3:::my_bucket/*"]
    }
  ]
}
```

AWS Storage Services

- Amazon S3 Bucket Security
 - IAM Policies
 - Specify what actions are allowed or denied on what AWS resources
 - Attached to IAM users, groups, or roles

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": ["arn:aws:s3:::my_bucket",
                 "arn:aws:s3:::my_bucket/*"]
  }]
}
```


AWS Storage Services

- Amazon S3 Bucket Security
 - Access Control Lists
 - Best practice to use S3 bucket policies or IAM policies
 - S3 ACL is a legacy method
 - Can be attached to individual objects within a bucket to manage permissions for those objects





AWS Storage Services

- Amazon S3 Bucket Security
 - IAM Policies vs Bucket Policies
 - Use IAM if
 - You need to control access to AWS services other than S3
 - Many S3 buckets with different permissions
 - Use Bucket Policies if
 - For simple cross-account access to your S3 environment
 - You prefer to keep access control policies in the S3 environment



AWS Storage Services

- Amazon S3 Bucket Security
 - The Principle of Least Privilege!
 - When using multiple authentication mechanisms, the decision is made using a union of all IAM policies, S3 bucket policies, and S3 ACLs
 - Deny and Explicit Deny always trumps an Allow
 - If IAM policy grants access to object but a bucket policy denies access, the deny takes precedence



AWS Storage Services

- Amazon S3 Encryption – Data at Rest
 - Server-side encryption using
 - Amazon S3-Managed Keys (SSE-S3)
 - AWS KMS-Managed Keys (SSE-KMS)
 - Customer-Provided Keys (SSE-C)
 - Client-side encryption using
 - AWS KMS-managed customer master key
 - Client-side master key



AWS Storage Services

- Amazon S3 Encryption – Data at Rest
 - Amazon S3-Managed Keys (SSE-S3)
 - Set in the console or via API flag
 - Each object encrypted with unique data key
 - Key is then encrypted with a rotated master key managed by Amazon S3
 - AES-256 is used for both object and master key
 - **x-amz-server-side-encryption** header set to SSE-S3



AWS Storage Services

- Amazon S3 Encryption – Data at Rest
 - AWS KMS-Managed Keys (SSE-KMS)
 - Define AWS KMS master key within each account
 - Master key then used to encrypt the data key
 - Request to create object key created when upload starts
 - KMS creates object key and encrypts with master key
 - **x-amz-server-side-encryption** header set to SSE-KMS
 - **x-amz-server-side-encryption-aws-kms-key-id** is used to indicate ID of master encryption key

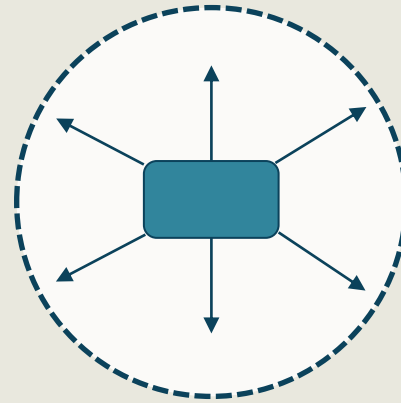


AWS Storage Services

- Amazon S3 Encryption – Data at Rest
 - AWS SSE-C (Customer-provided keys)
 - Use your own encryption key when uploading to S3
 - AES-256
 - Encryption key is deleted from AWS S3
 - Must provide same key in request when retrieving data
 - **x-amz-server-side-encryption-customer-algorithm** header set to Yes when SSE-C used

AWS Storage Services

- Amazon S3 Encryption – Data in Transit
 - SSL API endpoints protection for data in transit
 - Client-side encryption protects data in transit because the objects are encrypted before being sent to AWS





AWS Storage Services

- Amazon S3 Storage Classes
 - S3 Standard:
 - General purpose storage of frequently accessed data
 - Object data stored redundantly across multiple AZs
 - S3 ONEZONE_IA:
 - Stored in only one AZ
 - Less expensive but not as resilient
 - S3 Intelligent Tiering
 - Great for cost optimization
 - Moves data between frequent and infrequent access when patterns change



AWS Storage Services

- Amazon S3 Bucket Security
 - Access Control Lists
 - Best practice to use S3 bucket policies or IAM policies
 - S3 ACL is a legacy method
 - Can be attached to individual objects within a bucket to manage permissions for those objects



AWS Storage Services

- Amazon S3 Glacier
 - Used for archiving data long-term
 - Not available for real-time access, must be restored before accessing
 - Retrieval Options:
 - Expedited: Data available in 1-5 minutes
 - Standard: Data available in 3-5 hours (Default option)
 - Bulk: Data available in 5-12 hours



AWS Storage Services

- Amazon S3 Glacier Deep Archive
 - Used for long term retention of data that is rarely accessed in a year
 - Lowest cost
 - Replicated and stored across 3 Azs
 - 12 hour or less restoration
 - Bulk retrieval: Petabytes of data in 48 hours



AWS Storage Services

- Amazon S3 Multit-Part Upload
 - Allows you to upload large objects in multiple parts
 - Higher throughput: Uploading in parallel
 - Easier error recovery: Only re-upload the parts that failed
 - Pause and Resume uploads
 - Upload objects as you create them



AWS Storage Services

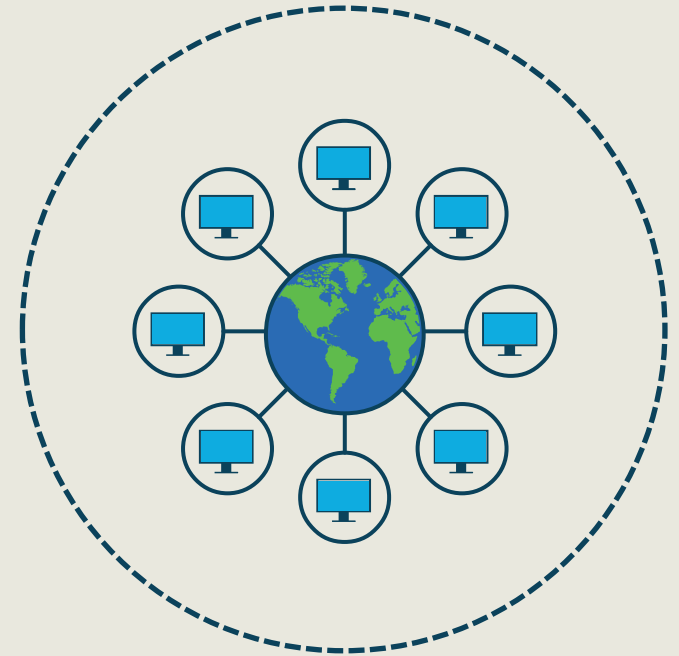
- Amazon S3 Select
 - Used to query only the data you need from an object; used to improve performance
 - Works on objects stored as CSV or JSON
 - Integrated with CloudWatch metrics

AWS Storage Services

- Amazon CloudFront
 - Amazon's Content Delivery Network (CDN)
 - Speeds up distribution of static and dynamic web content
 - DDoS protection, integrated with AWS Shield and AWS WAF
 - Content is sent to edge locations; when user requests content served on CloudFront, user goes to edge location with lowest latency
 - If content is not in that edge location, it is retrieved from the origin (S3 bucket, HTTP server)
 - Can set up HTTPS communication between the viewers and CloudFront by:
 - Setting the **Viewer Protocol Policy** to use **Redirect HTTP**
 - Set the **Viewer Protocol Policy** to **HTTPS** only

AWS Storage Services

- Amazon CloudFront - Origins
 - S3 Bucket
 - Distribute static files/websites and cache at edge locations
 - Security using CloudFront Origin Access Identity (OAI)
 - Can also use CloudFront to upload files to S3
 - Custom Origin(HTTP)
 - EC2 instance, ALB, S3 website
 - Any custom HTTP backend



AWS Storage Services

- Amazon CloudFront – Geo Restriction
 - CloudFront is very useful in restricting access to your content via whitelisting and blacklisting countries
 - A 3rd party Geo-IP database is used to determine the users country
 - This could be used to control access to copyrighted content



AWS Database Services



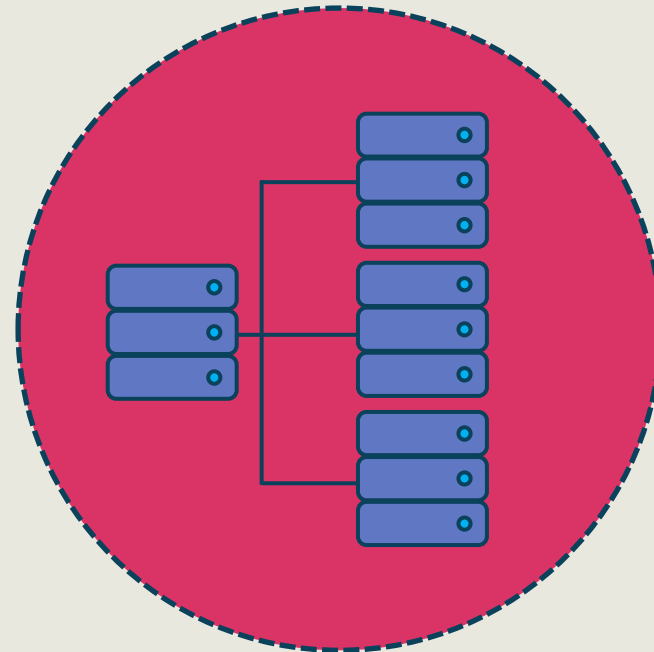


AWS Database Services

- AWS Relational Databases
 - Amazon RDS
 - Managed Relational Database Service
 - You don't worry about hardware, setup, scaling, replication, and many other management tasks
 - Fully featured database without the database administration
 - Unmanaged relational databases
 - Hosted on an EC2 instance
 - More flexibility on types of databases
 - You are responsible for administration

AWS Database Services

- Amazon RDS
 - Billed via DB instance hours
 - Backups, software patching, failure detection and recovery is all managed
 - Supports:
 - Aurora
 - MySQL
 - MariaDB
 - PostgreSQL
 - Oracle
 - Microsoft SQL Server





AWS Database Services

- Amazon RDS
 - Made up of a DB instance
 - Can have up to 40 Amazon RDS instances
 - Each DB instance runs a DB engine
 - Computation and memory capacity is determined by a changeable DB instance class
 - Standard
 - Memory Optimized
 - Burstable Performance



AWS Database Services

- Amazon RDS High Availability
 - Primary instance with synchronous secondary instance as failover in different AZ
 - Read replicas are available to scale database reads
 - Asynchronous copy from DB instance to read replica
 - Route read queries to the Read Replica
 - Can create Read Replica that has a different storage type from the source DB
 - Cross-region Read Replicas in multi-regions
 - Cannot have encrypted Read Replica of unencrypted DB, or vice versa



AWS Database Services

- Amazon RDS Security
 - DB Security Groups
 - Used in EC2-Classical
 - Controls access to DB instance that is not in a VPC
 - VPC Security Groups
 - Used in EC2-VPC
 - Controls access to DB instance in VPC
- EC2 Security Group
 - Used in EC2, can be used with DB instance as well

AWS Database Services

- Amazon RDS Security – Best Practices
 - Never use root account; individual IAM accounts only for DBAs
 - Principle of Least Privilege!
 - Rotate IAM credentials
 - Security Groups are useful to control what IP addresses or EC2 instances can connect to the DB
 - Run inside a VPC to take advantage of NACLs
 - RDS encryption will protect your data at rest
 - Managed Keys use AWS KMS (Region specific)
 - SSL connections will protect your data in transit



AWS Database Services

- Amazon RDS Storage Auto Scaling
 - RDS automatically scales database storage capacity in response to workloads
 - Continuous monitoring of storage consumption and auto scaling when utilization gets too close to provisioned capacity



AWS Database Services

- Amazon RDS Monitoring
 - Amazon CloudWatch Metrics and Alarms
 - RDS automatically sends metrics to CloudWatch every minute for every active database
 - CPU utilization, Connections, IOPS (Enhanced), Latency (Enhanced), Throughput (Enhanced), Queue Depth (Enhanced)
 - Amazon CloudWatch Logs
 - Can send your DB logs to CloudWatch Logs in order to monitor, store and access the logs
- RDS Events
 - Notification when changes occur to a DB instance, snapshot, parameter group or security group
 - Incorporate with SNS for instant notification



AWS Database Services

- Amazon Aurora
 - Managed PostgreSQL and MySQL compatible relational database
 - Performance and high availability (5x throughput of MySQL, 3x throughput of PostgreSQL)
 - Simple and cost effective
 - Can reuse tools, code, and applications that you currently use with MySQL and PostgreSQL
 - Automatic scaling up to 64 TB
 - Shareable to other AWS accounts for cloning
 - Fault-tolerant and self-healing
- Amazon DynamoDB Time to Live (TTL) allows you to define a per-item timestamp to determine when an item is no longer needed

AWS Database Services

- Amazon Aurora – DB Clusters
 - **DB Cluster** is one or more DB instance and a cluster volume to manage the data on the instances
 - A **cluster volume** is virtual database storage volume, spread across multiple AZs, each holding a copy of the data
 - Cluster Types:
 - Primary DB Instance
 - 1 per cluster
 - Performs all data modifications to cluster volume
 - Aurora Replica
 - Supports only Read operations
 - Up to 15 replicas (synchronous replication)



AWS Database Services

- Amazon Non-relational (NoSQL) Databases
 - Unlike relational databases, NoSQL databases use dynamic schemas for unstructured data
 - Easy to use, low latency
 - Optimized for applications that have large amounts of data and flexible data models
 - Use different methods for data management such as in-memory key value stores, and graph data models vs rows and columns
 - Use Cases:
 - Internet of Things (IoT)
 - Gaming applications
 - Mobile applications
 - Online transaction processing (OLTP)

AWS Database Services

- Amazon Non-relational (NoSQL) Databases

Type	Relational Databases	NoSQL Databases
Data Storage	Rows/Columns	Key-value, document, wide-column, graph
Querying	SQL Queries	Collection of documents
Scalability	Vertical	Horizontal
Schemas	Fixed	Dynamic
Consistency	Strong	Eventual and Strong
Transactions	Supported	Support varies



AWS Database Services

- Amazon DynamoDB
 - Amazon's fully managed NoSQL database offering
 - Supports key-value and document stores
 - Can store and retrieve any amount of data
 - Can serve any level of traffic
 - On-demand backups
 - Point in time recovery (up to 35 days)
 - Automatically replicated across multiple AZs
 - Full support for multi-master writes
 - Supports Endpoints with IP addresses for routing and firewall policies

AWS Database Services

- Amazon DynamoDB Components

- **Tables**

- Where data is stored; collection of data
 - No set schema
 - 256 tables per region

- **Items**

- A collection of attributes
 - **Primary Keys** are used to uniquely identify each item
 - **Secondary indexes** are used to improve querying and scanning





AWS Database Services

- Amazon DynamoDB Components
 - **Attributes**
 - Each item is made up of one or more attributes
 - Similar to fields or columns in a RDS
 - Ie. Item in Customers table have attributes like CustomerID

AWS Database Services

- Amazon DynamoDB Components
 - **Primary Keys**
 - Uniquely identifies each item in a table
 - Required; Must be unique
 - DynamoDB supports two types of primary keys:
 - Partition Key (hash key)
 - Partition Key and sort key



AWS Database Services

- Amazon DynamoDB Components
 - **Partition Key (hash key)**
 - Single primary key with one attribute known as partition key
 - Value is used as input to a hash function. Output of that function determines where the data is stored
 - If table only has partition key (no sort key), no two items can have same partition key value
 - **Partition Key and sort key (composite primary key)**
 - Composed of partition key and a sort key
 - Sort key is also referred to as the range attribute
 - Two items can have the same partition key if a sort key is used on the table
 - I.e. *Movies* table has a composite primary key (*Title* and *Year*), you can access any items in the *Movie* table if you provide the *Title* and *Year* values for the item
 - More flexibility in querying. Can only look for movies in a specific year if we wish

AWS Database Services

Partition Primary Key



IoTID (Partition Key)	Street	Pole
IOT1	Maple	3
IOT2	Chanterelle	5

Partition and Sort Primary Key



IoTID	Time	Value
IOT1	12-20-2020-14:20:20	5
IOT2	12-21-2020-16:15:20	12
IOT1	12-21-2020-18:12:23	3



AWS Database Services

- Amazon DynamoDB Components
 - **Secondary Indexes**
 - Enables you to query data in a table using an alternate key in addition to the primary key
 - Can create one or more secondary indexes
 - Two types of indexes:
 - **Global secondary index:** Index with partition key and sort key that can be different from those on the table. Define up to 20
 - **Local secondary index:** Index that has same partition key as the table, but different sort key. Can define up to 5 local secondary indexes

Using a *Movies* table, we can query data by *Title* (Partition key), or by *Title* and *Year* (Partition key and sort key).

If we want to query data by *LeadActor* and *Genre* we need to create a global secondary index on *LeadActor and Genre*.

AWS Database Services

- Amazon DynamoDB GSI vs LSI

Characteristic	GSI	LSI
Query Scope	Entire table, all partitions	Single partition
Key attributes	Partition Key, or Partition Key and Sort Key	Partition and Sort Key; Partition key must be same attribute as base table
Read Consistency	Eventual consistency only	Eventual or strong consistency

AWS Database Services

- Amazon DynamoDB Provisioned Mode
 - Differs from On-Demand as we are “Provisioning” our needs
 - When reading data from a table, the response is **eventually consistent**; you may have stale data.
 - If you request a **strongly consistent read**, you get the most up to date data.
 - **Strongly consistent reads** are not available across AWS regions
 - Throughput requirements must be specified when creating a table or index in DynamoDB
 - Specified in **Read Capacity Units (RCU)** and **Write Capacity Units (WCU)**

**** VERY IMPORTANT ****

AWS Database Services

- Amazon DynamoDB RCUs
 - 1 RCU = 1 **Strongly Consistent Reads** per second for an item up to 4kb in size
 - 1 RCU = 2 **Eventually Consistent Reads** per second for an item up to 4kb in size
 - 2 RCU = 1 **Transactional read request (one read per second)** up to 4kb in size
 - How do we calculate our RCU needs if data has average size of 3.5KB and application will send 150 eventually consistent read requests per second?
 - Step 1: Get the average item size by rounding up to 4kb
 - 3.5KB = 4KB rounded up
 - Step 2: Get the RCU per Item by dividing the average item size by 8Kb (because eventually consistent)
 - $4KB / 8KB = 0.5$
 - Step 3: Multiply the RCU per item to the number of items per second
 - $150 \times 0.5 = 75$ eventually consistent read requests

AWS Database Services

- Amazon DynamoDB RCUs
 - How do we calculate our RCU needs if data has average size of 3.5KB and application will send 150 strongly consistent read requests per second?
 - Step 1: Get the average item size by rounding up to 4kb
 - 6.5KB = 8KB rounded up
 - Step 2: Get the RCU per Item by dividing the average item size by 4kb because strongly consistent)
 - $8\text{KB} / 4\text{KB} = 2$
 - Step 3: Multiply the RCU per item to the number of items per second
 - $150 \times 2 = 300$ strongly consistent read requests

AWS Database Services

- Amazon DynamoDB WCUs
 - 1 **Write Capacity Unit (WCU)** = 1 write per second for an item up to 1kb in size (standard)
 - 2 **Write Capacity Units (WCU)** = 1 transactional write request (one write per second) up to 1kb in size
 - How do we calculate our WCU needs if data has average size of 6KB and we anticipate 100 writes per second (standard writes)
 - Step 1: Item size x writes per second
 - $6\text{kb} \times 100 \text{ (writes per second)} = 600 \text{ Write Capacity Units}$
 - How do we calculate our WCU needs if data has average size of 6KB and we anticipate 100 writes per second (transactional)
 - Step 1: Item size x writes per second x 2 (2 WCUs = 1 transaction write)
 - $6\text{kb} \times 100 \text{ (writes per second)} \times 2 = 1200 \text{ Write Capacity Units}$

AWS Database Services

- Amazon Lambda Integration with DynamoDB streams
 - Lambda is a serverless offering from AWS which allows you to run code based on “triggers”
 - *DynamoDB* is one of these triggers
 - This allows us to build applications that will react to data changes in the DynamoDB tables
 - Lambda polls your DynamoDB streams and invokes your function synchronously when a stream is detected
 - Use Case:
 - Online learning platform using Lambda, Elastic Beanstalk and DynamoDB. Whenever a customer is added to the table, we want a Lambda function to send them a welcome email .
 - By configuring a DynamoDB stream on the table, we can configure it as an event source for our Lambda function that will send the email



AWS Database Services

- Amazon DynamoDB Streams
 - Optional features used to capture data modification events in DynamoDB tables
 - Real-time stream with data about these events
 - Each event is a stream record and is written whenever:
 - A new item is added to the table
 - An item is updated
 - An item is deleted from the table
 - Stream contains table name, timestamp, and have lifetime of 24 hours; then removed

AWS Database Services

- Amazon DynamoDB Accelerator (DAX)
 - Fully managed, in-memory cache for DynamoDB
 - 10x performance increase. Milliseconds to microseconds
 - Developers need not manage cache, data population, or cluster management
 - Enable via console or using the AWS SDK
 - Pay for capacity you provision



AWS Database Services

- Amazon ElastiCache

- Distributed **in-memory cache**
- Redis and Memcached engines to choose from
- Great for storing session state!

- **ElastiCache Nodes**

- The node is a fixed-size chunk of secure, network RAM
- Multiple nodes make a cluster and are of the same instance type and runs the same cache engine
- Each cache node has its own DNS name and port



AWS Database Services

- Amazon ElastiCache - Redis
 - Flawless transition for applications already using Redis
 - Automatic detection and cache node failure recovery
 - Multi-AZ with failover of a failed primary cluster, to a read replica
 - A shard is a collection of one to six related nodes in a cluster. Can partition data across up to 250 shards
 - Shards are created to support replication of data across various nodes so the cache is fault tolerant
 - Redis Clusters are a logical grouping of one or more ElastiCache for Redis shards
 - For increased fault tolerance, have at least two nodes in a Redis cluster and have Multi-AZ with automatic failover enabled
 - Replicas use asynchronous replication with the primary node



AWS Database Services

- Amazon ElastiCache – Redis Endpoints
 - Single Node Redis (Cluster mode disabled)
 - Used for both read and writes
 - Multi-Node Redis (Cluster mode disabled)
 - Primary endpoint for all writes
 - Read endpoint points to read replicas
 - Redis (Cluster mode enabled)
 - Single configuration endpoint
 - Application connects to configuration endpoint and discovers the primary and read endpoints for each shard in the cluster



AWS Database Services

- Amazon ElastiCache – Memcached
 - Automatic detection and failure recovery
 - ElastiCache Auto Discovery for Memcached allows your applications to identify all the nodes in a cache cluster
 - ElastiCache node access is restricted to applications running on EC2 instances
 - Supports large nodes
 - No support for Multi-AZ failover or replication
 - No support for snapshots



AWS Database Services

- Amazon ElastiCache – Memcached
 - Memcached Cluster
 - Logical grouping of one or more ElastiCache nodes.
 - Supports up to 100 nodes per customer
 - Each node has its own endpoint
 - Cluster has endpoint called the configuration endpoint



AWS Database Services

- Amazon ElastiCache – Caching Strategies
 - Lazy Loading
 - Only loads data when necessary
 - Only requested data is cached
 - Any node failures are not fatal
 - Cache miss penalty
 - Extra time spent retrieving the data from the database after an unsuccessful request to the cache
 - Stale data
 - Data may have changed in between requests. Different version on the cache vs. the database



AWS Database Services

- Amazon ElastiCache – Caching Strategies
 - Write Through
 - Adds or updates data in the cache whenever data is written to the database
 - Data is always up to date
 - Write Penalty because every write invokes two trips, a write to the database and a write to the cache
 - Potential data missing
 - Newly created nodes do not contain all data. No way to get the data in until it is updated in the database
 - Unused data in cache
 - Cache churn
 - If records are updated often, cache will be as well

AWS Database Services

- Amazon ElastiCache – Redis vs Memcached

	Redis	Memcached
Persistence	Persistent storage. DB replacement	Purely caching; uses DB a data origin
Object Types	Complex; Hashes, Lists, Sets	Key-Value only
Scaling	Vertical; Read Replicas	Vertical and Horizontal
Multi-AZ	Supported. Automatic failover	Not supported
Backup and Restore	Supported	Not supported
Size	Up to 512MB per key	Up to 1MB per key
Pub/Sub capabilities	Yes	No



AWS Database Services

- Amazon Redshift
 - Fully managed data warehouse
 - Petabyte-scale
 - Can extend data warehouse queries to your data lake
 - Only supports Single-AZ deployments
 - A Redshift Cluster is made up of a set of **nodes**
 - A **leader node** and one or more **compute nodes**
 - Scale in or out by adding nodes



AWS CloudWatch

- Core AWS monitoring tool for resources and applications
- Display metrics and create alarms from those resources
- Integrate with SNS for email notifications
- Does not aggregate data across regions
- Integrated with EC2 auto-scaling
- Used to reduce mean time between failure (**MTBF**) and mean time to resolution (**MTTR**)
- Offers a **global**, customizable Dashboard that you can use to monitor resources in one single view



AWS CloudWatch Concepts

- **Namespaces** are containers for CloudWatch metrics
 - No default namespace
 - AWS/service naming convention
- **Metrics** are time-ordered data points sent to CloudWatch
 - Region-specific
 - 15 month life
 - Metric math allows you to query multiple metrics and use math expressions to create new time series based on them
 - Note: CloudWatch doesn't collect memory usage and disk space utilization by default until you install the CloudWatch agent in the instances



AWS CloudWatch Concepts

- **Alarms**

- Tracks a single metric over a specified period of time and based on the metric value, will perform a specific action
- Alarm States:
 - OK: Metric or expression is within the defined threshold
 - ALARM: Metric or expression is outside the defined threshold
 - INSUFFICIENT_DATA: Metric not available, or not enough data. Usually at start of the alarm

AWS CloudWatch Alarm Creation

- When you create an alarm you specify the following settings:
 - **Period:** Length of time to evaluate the metric. Expressed in seconds
 - **Evaluation Period:** Number of most recent data points to evaluate
 - **Datapoints to Alarm:** Number of data points within evaluation period that must be breached to cause the alarm to sound. Do not have to be consecutive, just within the last number of data points in the evaluation period

AWS CloudWatch Logs



Log aggregation



Log searches



Log processing



AWS CloudWatch

- Custom High Resolution Metrics
 - Publish via AWS CLI or API
 - Granularity at one second (1 minute for standard resolution metrics)
 - Much more immediate insight



AWS X-Ray

- Used to analyze and debug applications, specifically those deployed in a decoupled architecture
- Can be used to
 - Identify performance bottlenecks
 - Pinpoint specific service issues
 - Identify errors
 - Identify impact to users
- X-Ray integrates with AWS SDK and will add traces to track the application requests
- Easily integrate other AWS services with AWS X-ray (Lambda, API Gateway, ELB, EB)
- Uses trace data to generate a **detailed service graph**

AWS X-Ray



Trace

Path of a request

Segment

Data from a service

Subsegment

Identifies API calls

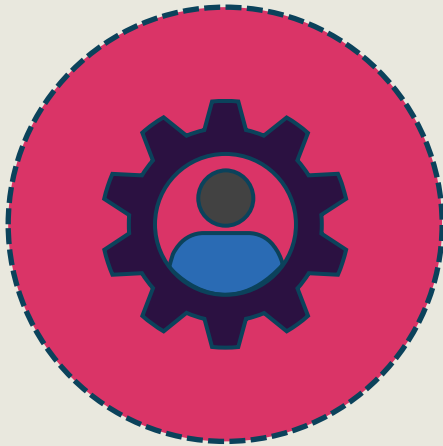


AWS X-Ray SDK

- The X-Ray SDK provides:
 - **Interceptors** are added to your code to trace incoming HTTP requests
 - **Client handlers** to instrument AWS SDK clients that your application will use to call other AWS services
 - **HTTP client** is added to call other internal and external HTTP web services
 - Supports applications in Node.js, Java, and .NET
 - SDK sends JSON segment documents to an X-Ray daemon that is listening for UDP traffic
 - The **X-Ray daemon** buffers segments in a queue and uploads them to X-Ray in batches
 - Enabled in Elastic Beanstalk by including the **xray-daemon.config** configuration file in the **.ebextensions** directory of your source code

AWS CloudTrail

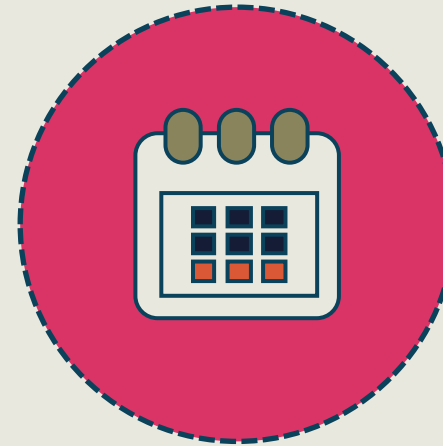
- API auditing service
- All actions taken by a user, role or service in the console, CLI, SDK, or APIs are recorded as **events**



Management events



Data events



Global service events



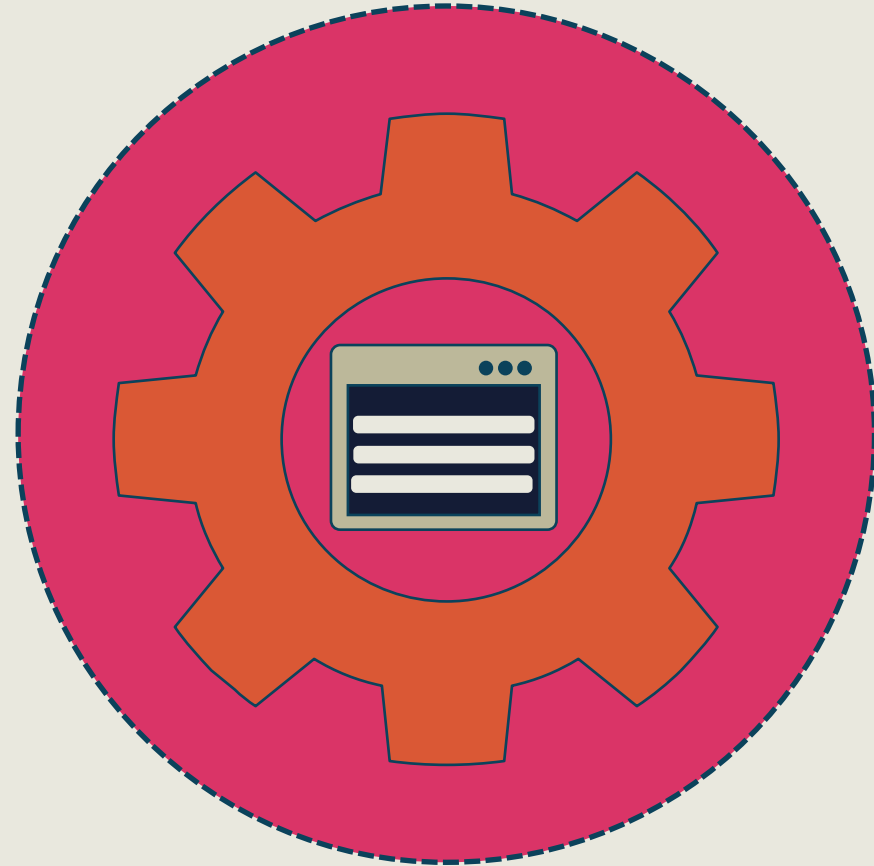
AWS CloudTrail Trails

- CloudTrail Trails are created to archive, analyze and respond to changes in AWS resources
- Types of Trails:
 - A trail that applies to **all regions**. This is the default trail when using console
 - A trail that applies to **one region**. This is the default trail when using CLI or API
 - **Organizational trail** will log events for all AWS accounts in an AWS Organization
 - CloudTrail event log files are encrypted using S3 SSE. You can use AWS KMS
 - Can implement SNS notifications
 - Can define S3 lifecycle rules to archive or delete log files
 - CloudTrail publishes log files every five minutes

AWS CloudFormation

Infrastructure as Code

- Helps remove dependency
 - Human intervention
- Deploy infrastructure
 - Declarative template syntax

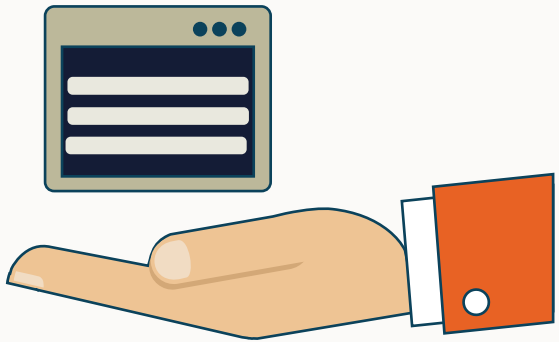


AWS CloudFormation

- Provisioning and managing
 - Cloud resources
 - Human readable
 - Machine consumable
- AWS CloudFormation
 - Template
 - Creating, updating, and deleting resources



Benefits of Infrastructure as Code



Scalability

Security

Stability

Transactional

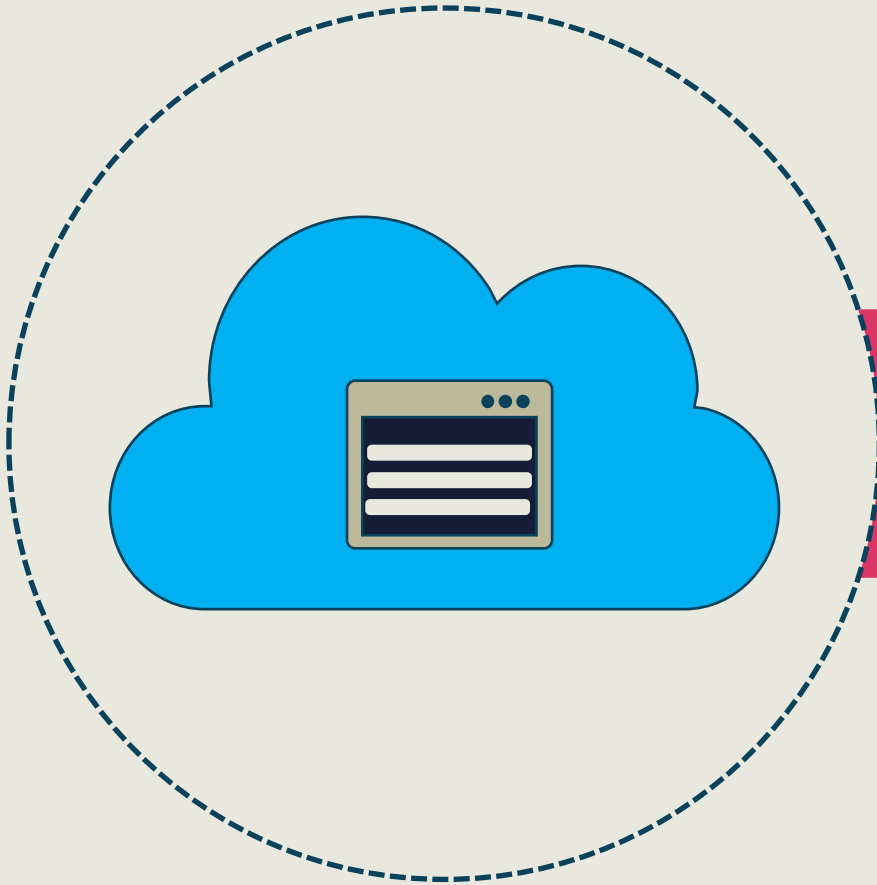
Visibility

AWS CloudFormation



- Infrastructure resources in cloud environment
 - Common language
- Simple text file
 - Model and provision
 - Automated and secure
 - Regions and accounts
- Available at no additional charge

AWS CloudFormation



Repeatable

Versionable

AWS CloudFormation

- Scalable web application
 - Back-end database
 - Auto scaling group
 - Elastic load balancer
- Create or modify existing template
 - Create stack



AWS CloudFormation

Replicate infrastructure

- Additional availability
- Multiple regions
- Reuse template
 - Consistently and repeatedly

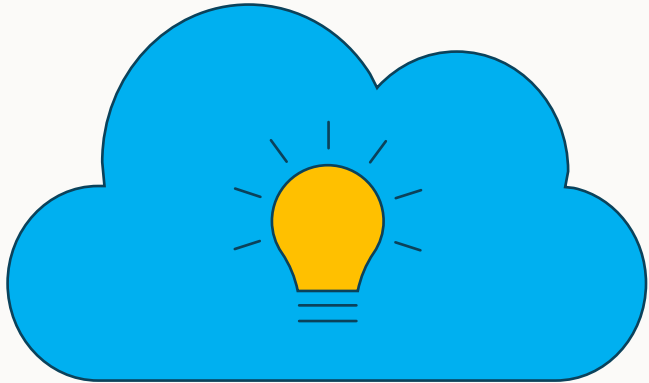


AWS CloudFormation

- Control and track changes
- Upgrade incrementally
- Roll back infrastructure
 - Original settings
- Version control system
 - Templates



AWS CloudFormation Concepts



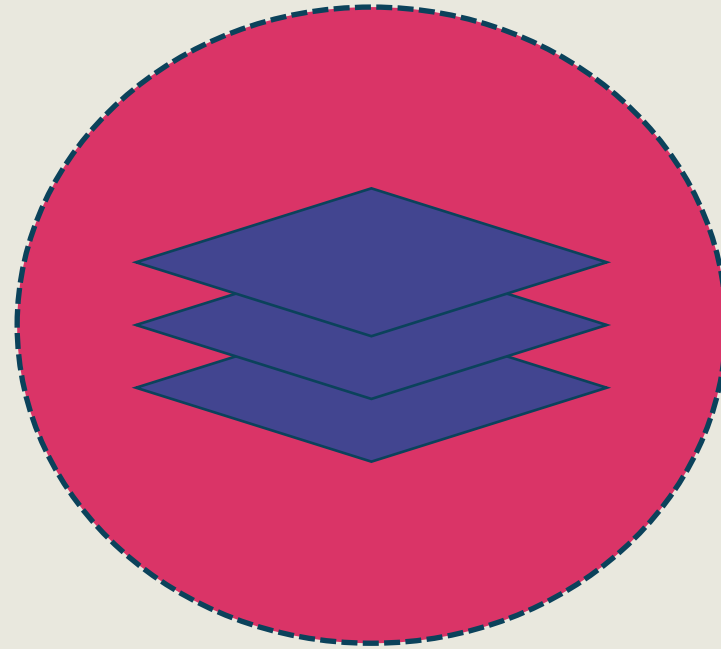
Stacks

Change sets

Templates

AWS CloudFormation Stacks

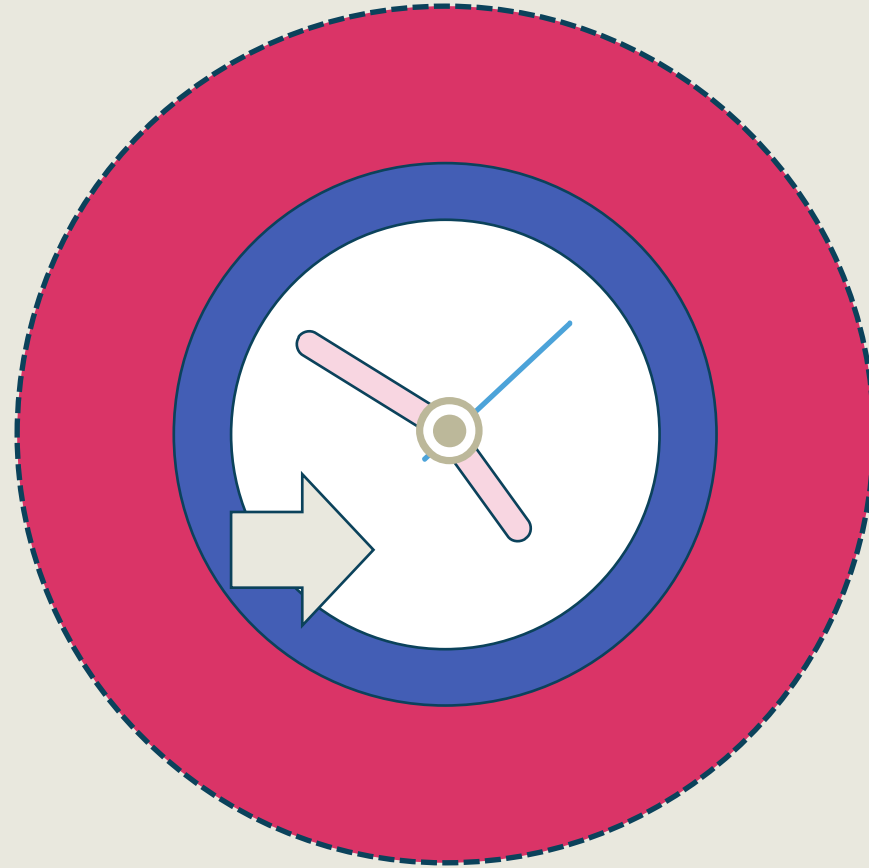
- Collection of resources
 - Template
- Modifications
 - Affect resources
- Create, update, delete



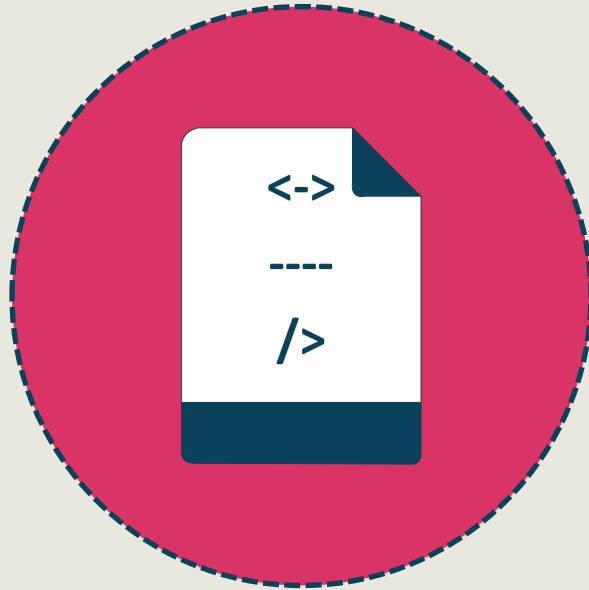
AWS CloudFormation Change Sets

Changes to resources

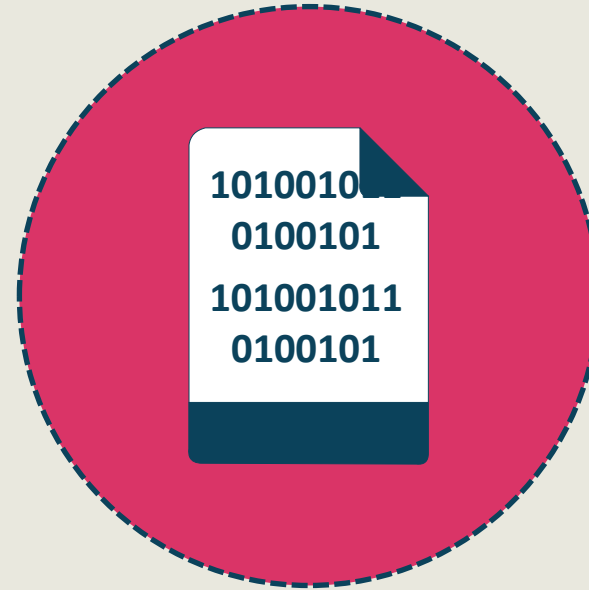
- Summary of proposed changes
- View impact of changes
 - Before implementing them



AWS CloudFormation Templates

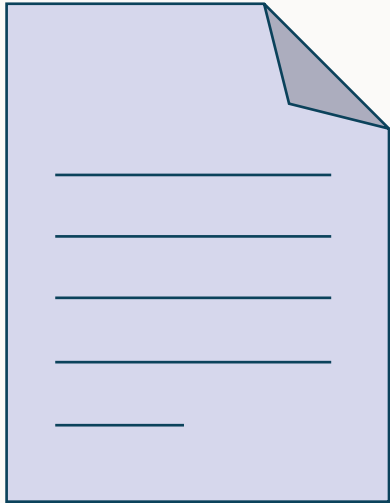


JSON



YAML

AWS Template Components



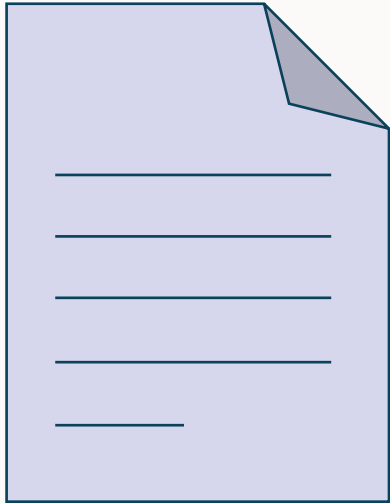
Description

Parameters

Mappings

Metadata

AWS Template Components



Resource

Conditions

Outputs

AWS CloudFormation Permissions

- IAM user or AWS role
 - Invoke a stack action
 - iam:PassRole permission
- Service role
 - Create, update, or delete actions
- Specify conditions
 - Control policies



AWS CloudFormation Transforms



AWS::Include Transform

- Import template snippets
 - Amazon S3 buckets

AWS::Serverless Transform

- Convert AWS SAM templates
 - AWS CloudFormation

AWS CloudFormation Intrinsic Functions

Fn::Base64	Fn::Join
Fn::Cidr	Fn::Select
Fn::FindInMap	Fn::Split
Fn::GetAtt	Fn::Sub
Fn::GetAZs	Ref



AWS CloudFormation Intrinsic Functions

Condition Functions

FN::AND

FN::EQUALS

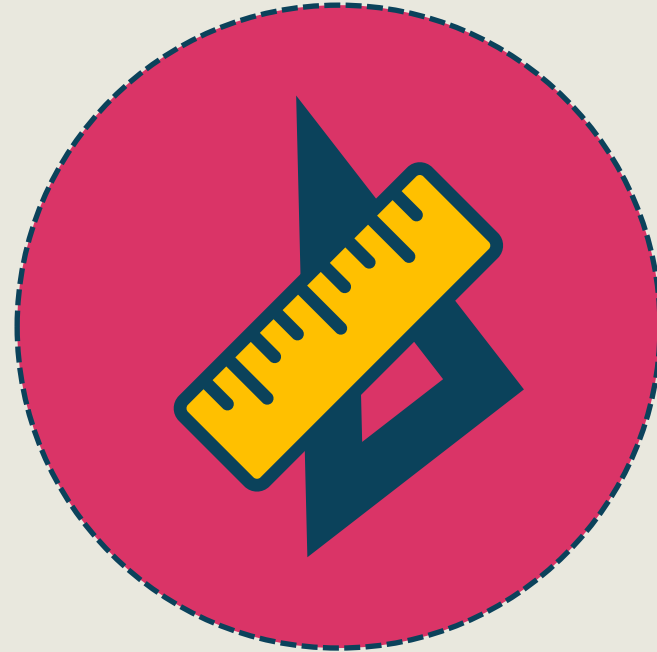
FN::IF

FN::NOT

FN::OR

AWS CloudFormation Designer

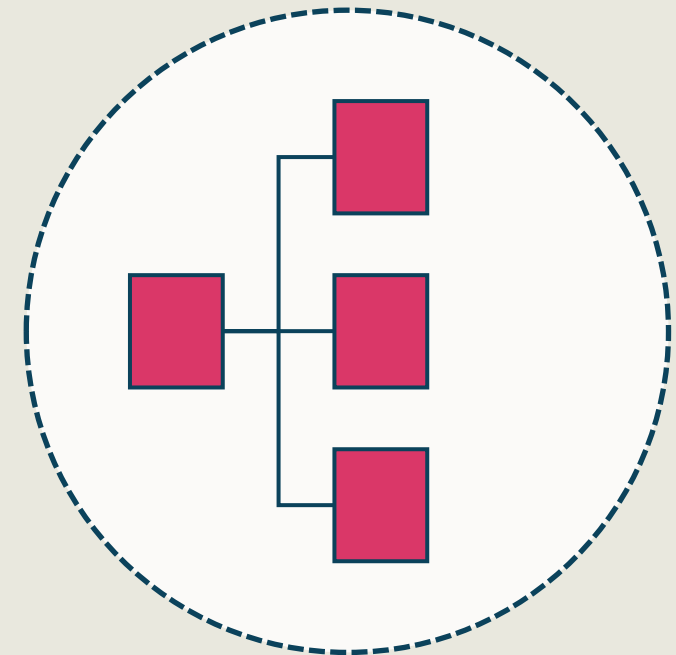
- Graphical interface
 - Design and deploy templates
 - Web-based
- Drag-and-drop
- Keeps track
 - Resource positions
 - Relationships
 - Metadata information



AWS Elastic Beanstalk

Instructor video will appear here during live session. Delete this placeholder before presenting

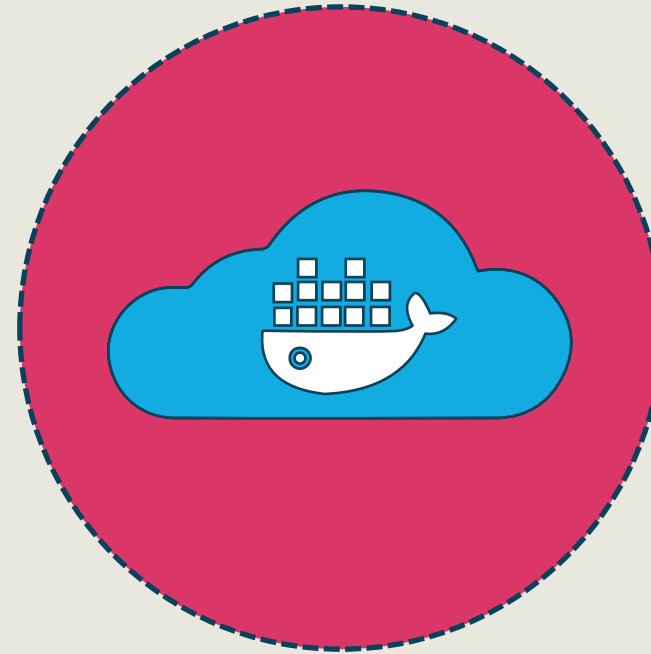
- Platform as a Service (PaaS)
 - Deploy and manage your applications without worry about the underlying infrastructure
 - Elastic Beanstalk does capacity provisioning, scaling
 - Supports the following languages:
 - Go
 - Java
 - .Net
 - Node.js
 - PHP
 - Python
 - Ruby



AWS Elastic Beanstalk

Instructor video will appear here during live session. Delete this placeholder before presenting

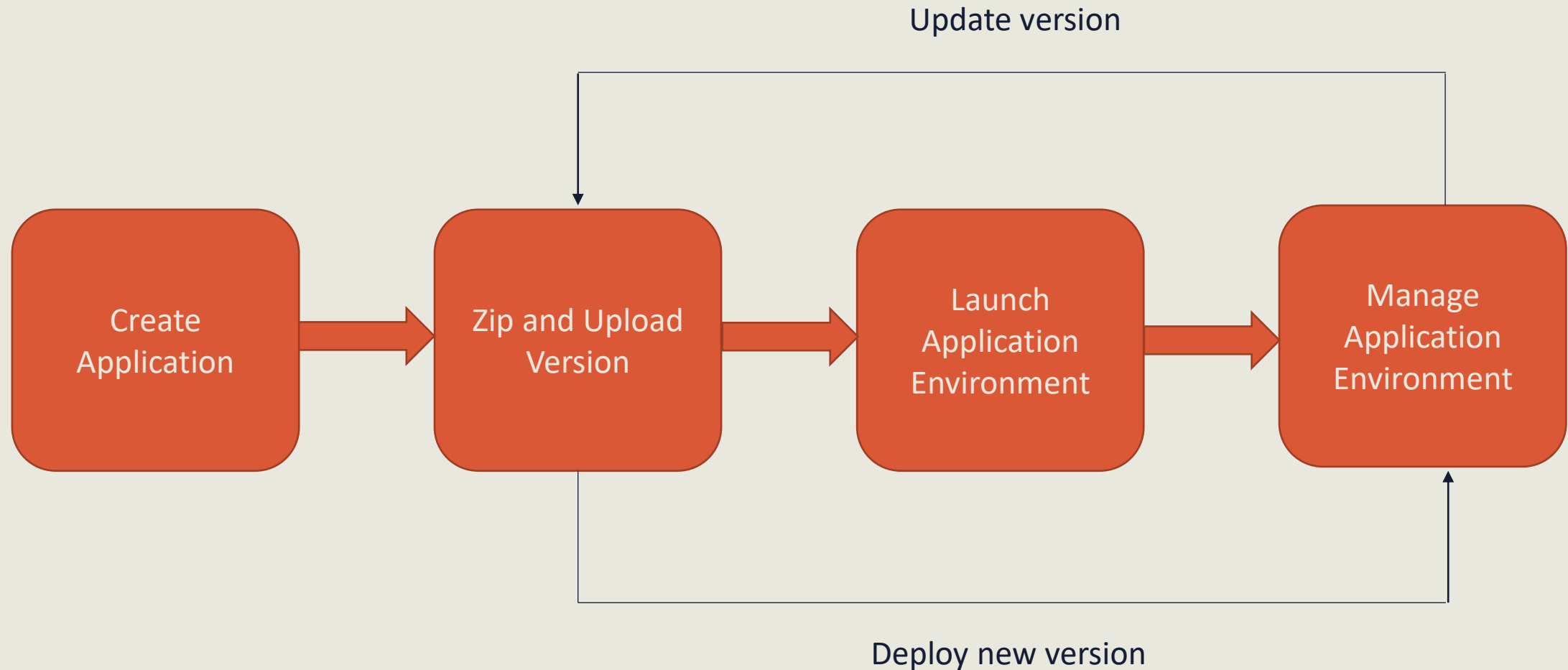
- Elastic Beanstalk
 - Supports web containers:
 - Tomcat
 - Passenger
 - Puma
 - Supports Docker containers



AWS Elastic Beanstalk

Instructor video will appear here during live session. Delete this placeholder before presenting

- Elastic Beanstalk Workflow



AWS Elastic Beanstalk

Instructor video will appear here during live session. Delete this placeholder before presenting

- Elastic Beanstalk Components

- **Application**

- collection of EB components; Environments, versions, and environment configurations
 - Think of it as a folder

- **Application Version**

- Specific, labeled version of code that makes up your web application
 - Points to an S3 object that contains the code
 - Applications can have many unique versions

- **Environment**

- Runs only one single applications version at a time





AWS Elastic Beanstalk

Instructor video will appear here during live session. Delete this placeholder before presenting

- Elastic Beanstalk Deployment Policies
 - **All at once**
 - New version is deployed to all instances simultaneously.
 - Involves Downtime!!
 - **Rolling**
 - New version is deployed in batches.
 - No downtime but reduced capacity as batches are taken offline and updated
 - **Rolling with additional batch**
 - New version is deployed in batches
 - New batch is launched first to ensure full capacity

AWS Elastic Beanstalk

Instructor video will appear here during live session. Delete this placeholder before presenting

- Elastic Beanstalk Deployment Policies
 - **Immutable**
 - Deploy to a fresh group of instances by performing an immutable update|
 - **Traffic Splitting**
 - Deploy to a fresh group of instances then split the incoming traffic between the existing application version and the new one
 - Blue/Green deployments ensure you have two production environments; “blue” and “green”. Once software working in green environment and testing is good, you switch the router so all requests go to the green environment and now the blue one stays idle. Makes for minimal downtime; Easy rollback
 - Canary deployments is similar to blue-green but less risky; uses a phased approach to roll out to all users

AWS Elastic Beanstalk

Instructor video will appear here during live session. Delete this placeholder before presenting

- Elastic Beanstalk Deployment Policies

Method	Impact
All at once	Downtime
Rolling	Single batch out of service
Rolling w/additional batch	If first batch fails, minimal
Immutable	Minimal
Blue/green	Minimal

AWS Elastic Beanstalk

Instructor video will appear here during live session. Delete this placeholder before presenting

- Elastic Beanstalk Command Line Interface (EB CLI)
 - Interactive commands for creating, updating, and monitoring your application
 - Used as an alternative to the Elastic Beanstalk console
 - Can be installed via:
 - EB CLI setup scripts from GitHub





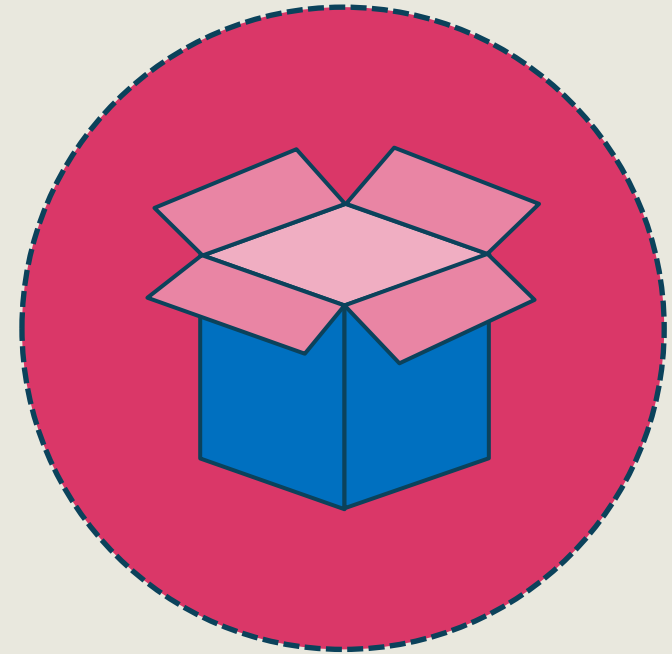
AWS Elastic Beanstalk

Instructor video will appear here during live session. Delete this placeholder before presenting

- Elastic Beanstalk Command Line Interface (EB CLI)
 - Configure your project directory
 - Create environments
 - `~/my-app4 eb create my-env`
 - Other commands:
 - `eb init application-name`: sets default values
 - `eb create`: creates new environment
 - `eb deploy`: deploys the application
 - `eb clone`: clones environment
 - `eb codesource`: used to configure EB CLI

AWS CI/CD

- Continuous Integration / Continuous Deployment
 - AWS CodePipeline
 - AWS CodeCommit
 - AWS CodeBuild
 - AWS CodeDeploy





AWS CI/CD

- Code Pipeline
 - Fully managed, **continuous delivery service** for automating release pipelines
 - A **pipeline** is made up of a series of **stages** (build, dev, test, prod). Each stage is comprised of a series of actions (tasks); this can be building code, or deploying to test environments
 - Pipelines must have at least two stages
 - Pipeline structure can be defined via a JSON document
 - A **revision** is a change made to the source location defined for the pipeline. Changes can include source code, data, configuration, or build output

AWS CI/CD

- AWS CodePipeline Workflow

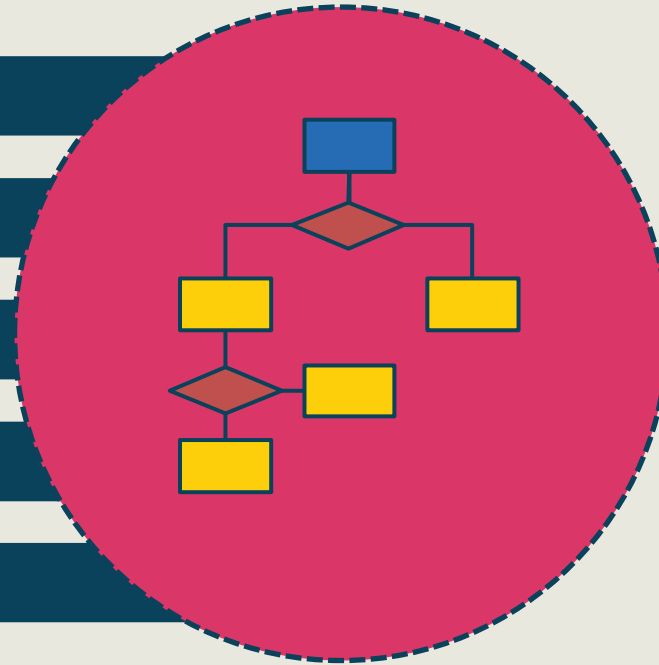
Source

Build

Stage

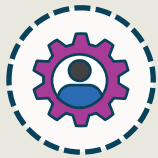
Approve

Deploy



AWS CI/CD

- AWS CodePipeline



Actions



Approvals



Approval actions



Artifacts



AWS CI/CD

- AWS CodeBuild
 - Fully managed, **code build service** used to compile source code, run tests, and produce build artifacts
 - A **build project** defines how a build is to be run. Contains information such as where the source code is located, which environment to use, commands to run, and where to store the output
 - A **build environment** is a YAML file that lets you specify what commands to run at each phase of the build
 - Must be named buildspec.yaml and placed in root of source directory
 - Supports Java, Python, Node.js, Ruby, Go, Android, .NET for linux, and Docker

AWS CI/CD

- AWS CodeBuild Options



Build timeout

Environment variables

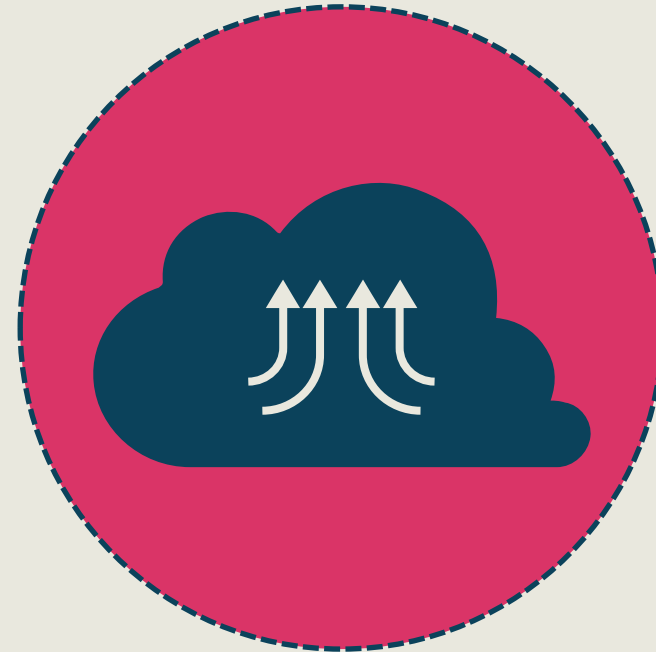
Output artifacts

Source branch

Source version

AWS CI/CD

- AWS CodeDeploy
 - Fully managed, **deployment service** for automatic software deployments to EC2, Fargate, Lambda, and On-Prem servers
 - Can be used to release new features
 - Avoid down-time!
 - Eliminate manual operations



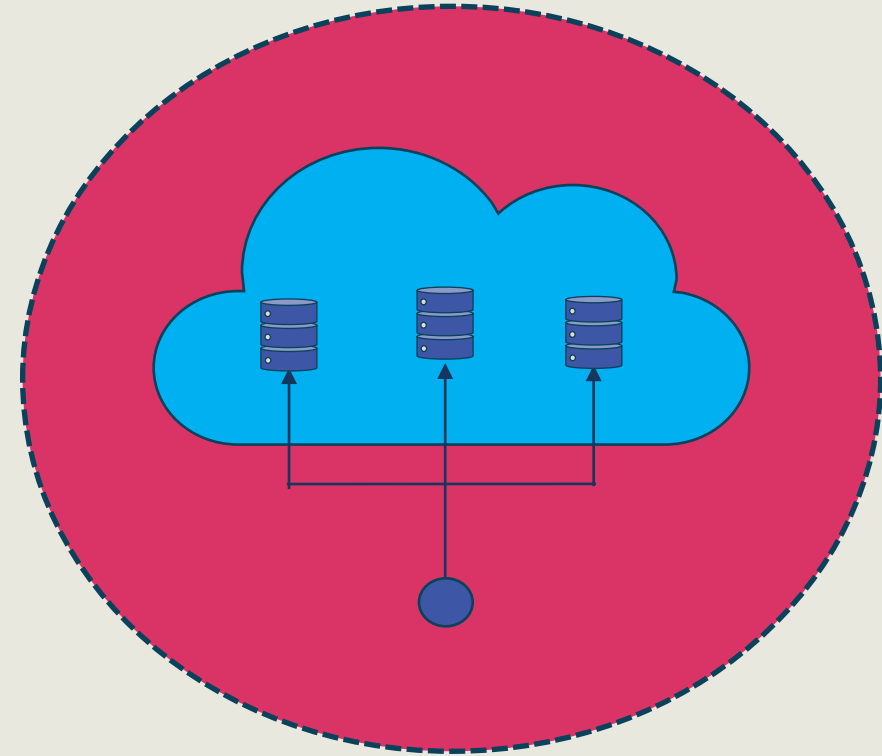


AWS CI/CD

- AWS CodeDeploy Concepts
 - **Application:** Unique name of application to deploy
 - **Compute Platform:** Platform to which application is deployed (EC2, Fargate, Lambda, On-Prem)
 - **Deployment Configuration:** Deployment rules and success/failure conditions
 - **Deployment Group:**
 - ECS: Specifies the ECS service, LB, test listener, and two target groups. Specifies when to route traffic to the replacement task set and when to terminate it
 - Lambda: CodeDeploy configurations for future Lambda function deployments
 - EC2/On-Prem: Set of individual instances targeted for deployment

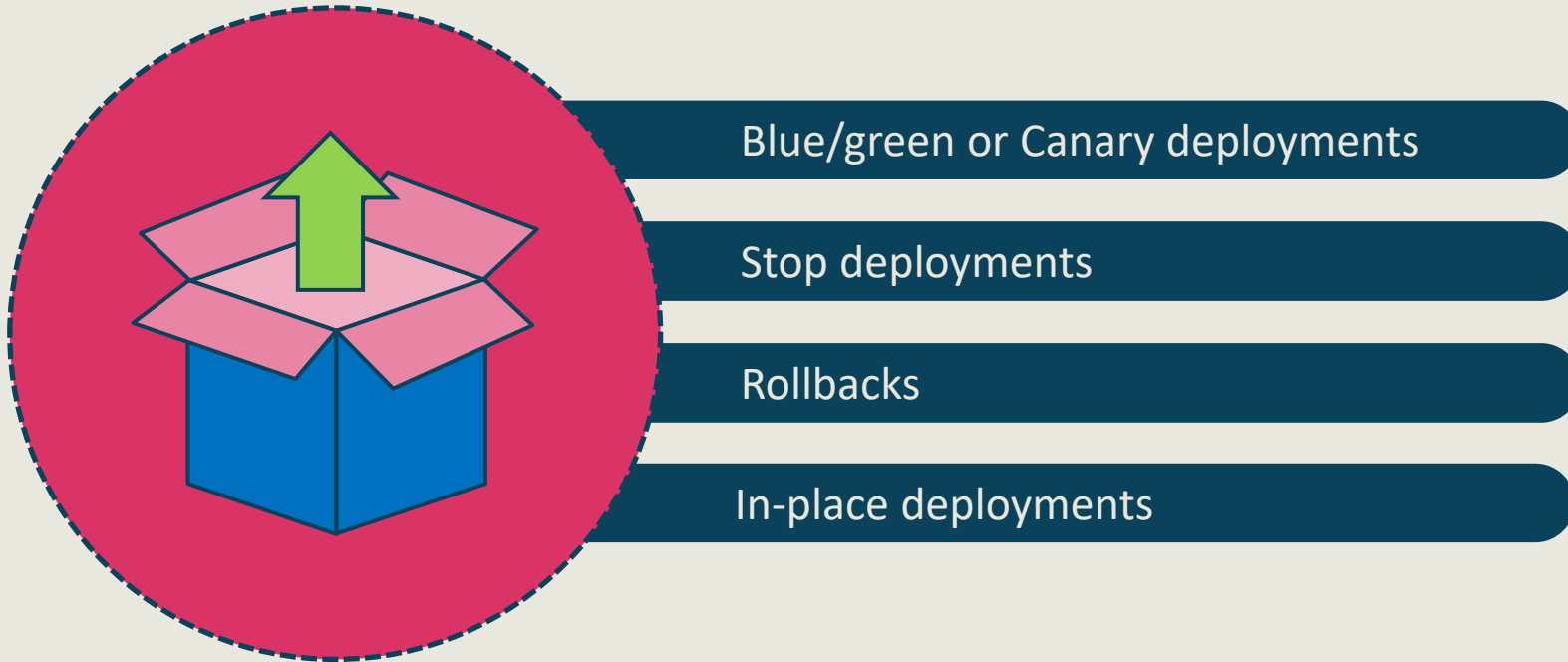
AWS CI/CD

- AWS CodeDeploy Features
 - Rolling updates
 - Deployment health tracking
 - Platform and language agnostic
 - Integrates with Amazon Auto Scaling, so new instances can have application deployed when spun up



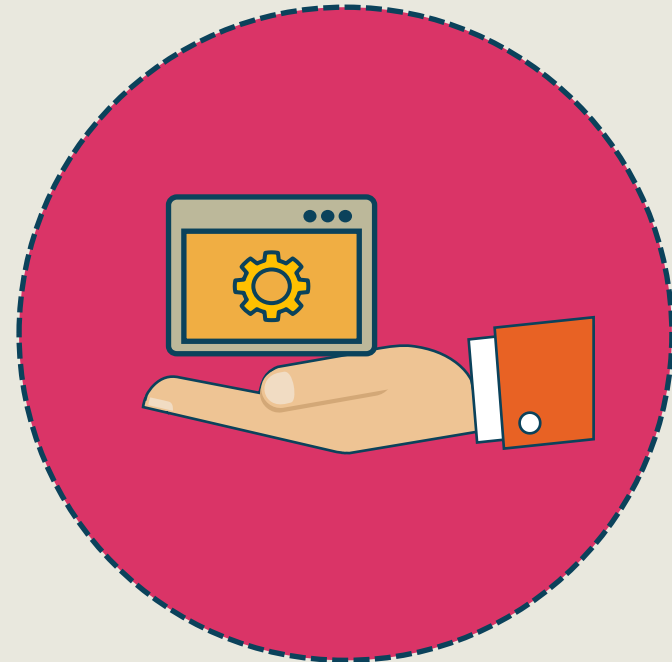
AWS CI/CD

- AWS CodeDeploy Deployments



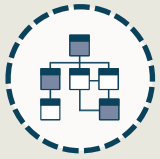
AWS CI/CD

- AWS CodeCommit
 - Fully-managed source control service similar to GitHub
 - Create own repository and use Git commands
 - Automatic encryption
 - Automatic scaling as required
 - Access control
 - Secure connectivity



AWS CI/CD

- AWS CodeCommit



Branches



Commits



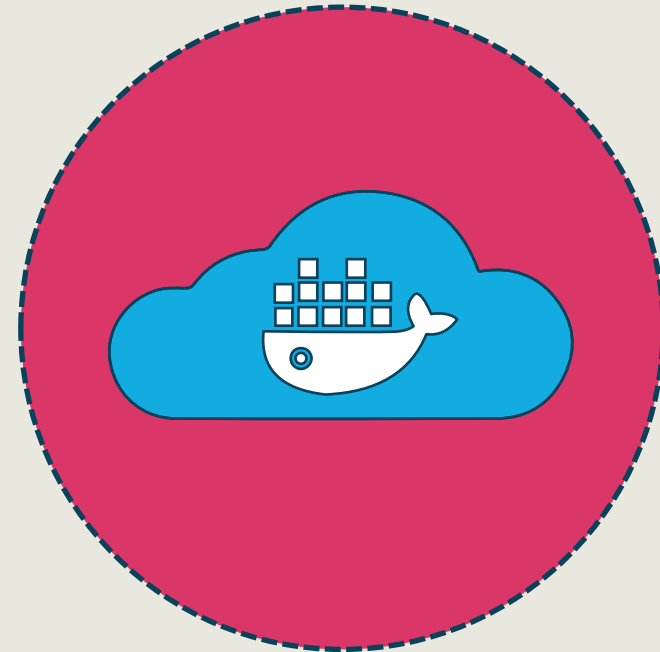
Files



Pull requests

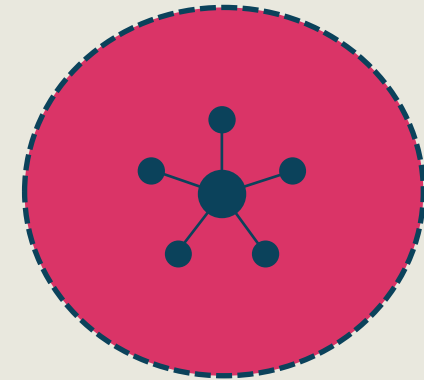
Amazon ECS

- Amazon ECS (Elastic Container Service)
 - Managed service to run, stop, and manage Docker containers
 - Create ECS clusters in a new or existing VPC
 - Define **task definitions** that specify which Docker container images to run across clusters
 - Streamlines managing and scheduling of containers
 - Highly scalable, high performance



Amazon ECS Clusters

- Foundational building blocks of ECS
- Simply just one or more EC2 instances in your VPC
- Each instance must have Cluster Agent installed to receive its scheduling and shutdown commands
- Clusters are region specific
- Used with EC2 and Fargate launch types
- If using Fargate, ECS manages your cluster resources
- If using EC2, you manage the cluster instances.
- **Cluster queries** are expressions that enable you to group objects based on specific attributes





Amazon ECS – Fargate

- Simpler method of managing containers in your environment
- You specify compute requirements, AWS automatically launches the instances to meet those requirements
- Cannot use EC2-Classic with Fargate (Must be launched in VPC)
- Only supports container images hosted on ECR or Docker Hub

Amazon ECS Tasks and Containers



Task definition

- JSON document
- Service-oriented architecture (SOA)

Containers

- Docker
- VMs share physical hardware



Amazon ECS Services

- Deployment Strategies
 - maximumPercent parameter
 - Defines the maximum percentage of tasks that is allowed to be running or pending
 - minimumPercent parameter
 - Defines the minimum percentage of tasks that is allowed to be running or pending
- Load Balancing
 - Classic
 - All tasks must be on same instance (Not good)
 - ALB
 - Route at application layer; dynamic host port mapping; path based routing
 - Network Load Balancer
 - Layer 4; High performance

Amazon ECS Schedule Tasks



Target tracking policies



Step scaling policies



Task placement strategies



Task placement constraints

Amazon ECS Service Discovery



Private image repositories

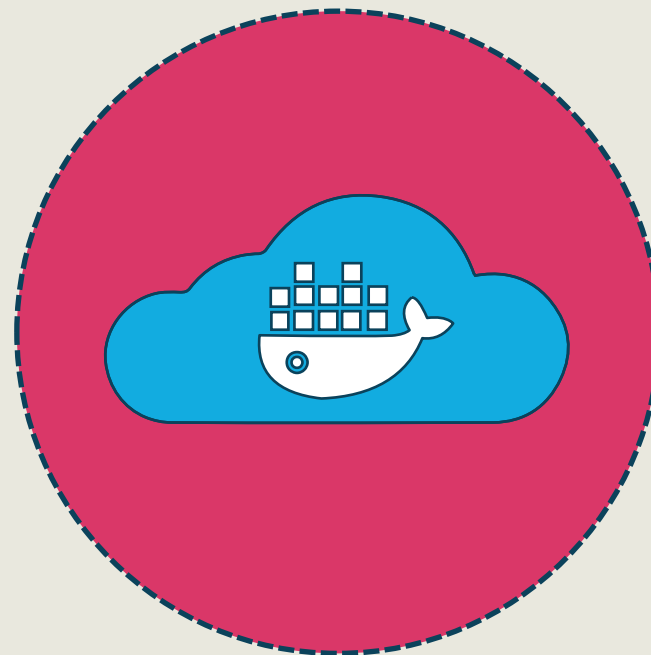
Elastic container repository

Container agent

Service limits

Amazon ECR

- Amazon ECR (Elastic Container Registry)
 - Managed AWS Docker registry service
 - Can use Docker CLI commands or Docker tools to maintain your workflow
 - ECR stores container images in S3
 - Can transfer container images to and from ECR via HTTPS



Amazon ECR Components

- **Registry**

- Provided to each AWS account; it's where we create image repositories and store our images
- Default URL is https://aws_account_id.dkr.ecr.region.amazonaws.com
- Authentication is necessary to use the registry
 - **Authorization token** needed to authenticate to ECR registries as an AWS user in order to push and pull images
 - *get-login* command on AWS CLI provides you with credentials to pass to Docker

- **Repository**

- Contains Docker images
- Resource based permissions are used to specify who can access the repository and what they can perform (**Repository policy**)
- Lifecycle policies allow us to specify the lifecycle of images in the repositories

Amazon ECR Authorization Tokens



Credential helper

Retrieve authentication token

HTTP API authentication



AWS OpsWorks Stacks

- Configuration management service to configure and operate applications by using **Puppet** or **Chef**
- Can manage in the cloud, or on-premises
- Can incorporate with load balancers, databases, and app servers
- Supports Chef Solo in order to install packages and frameworks, languages, and configuration software
- Ensures that your infrastructure adheres to standards

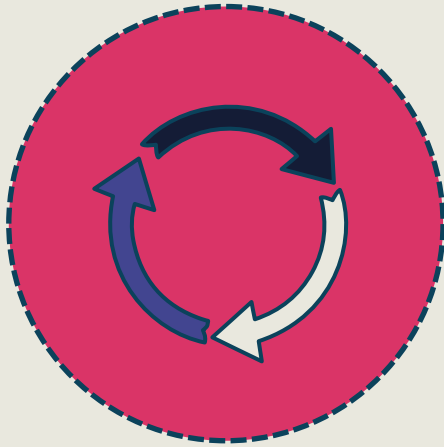
AWS OpsWorks Services

	Stacks	Chef Automate	Puppet Enterprise
Manage infrastructure	✓		
Chef	✓	✓	
Puppet			✓
Code repository		✓	✓
Built-in automatic scaling	✓		
Amazon EC2 Auto Scaling	✓	✓	✓
Compliance		✓	

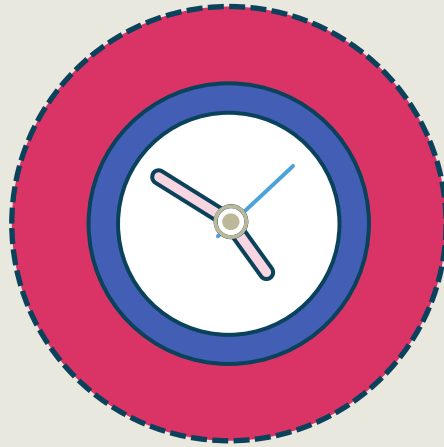
AWS OpsWorks Stack Concepts

- **Cookbooks**
 - Uses Chef to perform configuration (not in scope)
 - Cookbooks belong to a cookbook repository or a chef-repo.
 - A cookbook repository can hold one or more cookbooks
- **Recipes**
 - Actual configuration we want to implement
 - Uses nodes to execute in a run list
 - Execution is done in order the commands appear in the recipe

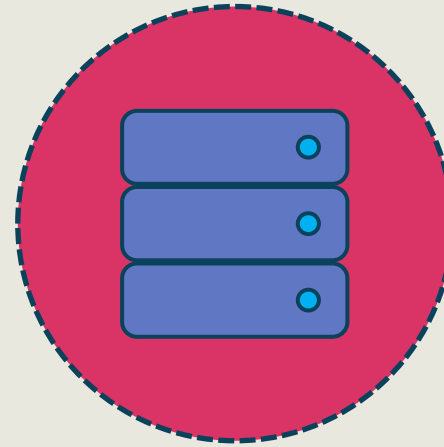
AWS OpsWorks Instances



24/7 instances

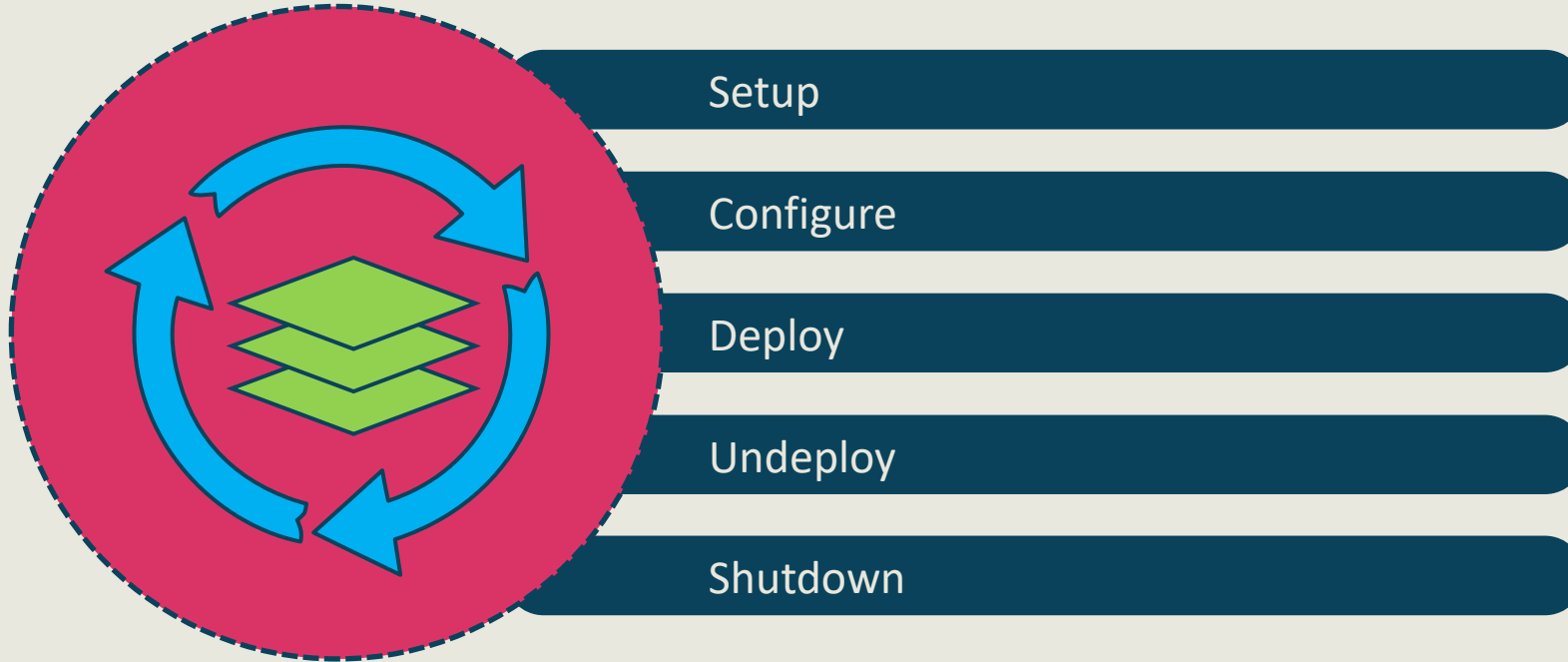


Time-based instances



Load-based instances

AWS OpsWorks Lifecycle



AWS Simple Queue Service (SQS)

- Hosted queue for decoupling distributed software systems and components
- Supports both **standard** and **FIFO queues**
- **Delay queues** allow you to delay the delivery of a new messages for a defined period of seconds
- **Visibility timeout** is a period of time SQS prevents other consumers from receiving and processing messages
- Uses a pull based polling method
- Can be accessed via VPC endpoints, without the need of public IPs and without having to travel the internet

AWS SQS Benefits



Availability

Durability

Reliability

Scalability

Security

AWS Simple Queue Service (SQS)

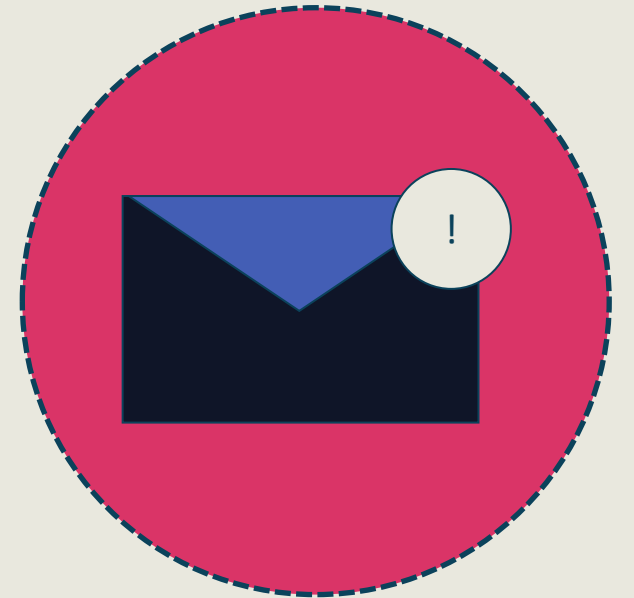
- Types of Queues
 - **Standard Queues**
 - Unlimited Throughput
 - At-least-once Delivery
 - Best-Effort Ordering
 - Use when throughput is important
 - **FIFO Queues**
 - High Throughput (3,000 msgs/s)
 - Exactly-Once Processing
 - First-in First-Out Delivery
 - Use when order of events is important

AWS SQS Standard Queues

- Default option
- Consumes messages using **short polling** by default.
 - ReceiveMessage request queries only a subset of servers (based on WRD) to find messages.
 - Short polling is implemented when *WaitTimeSeconds* parameter of a *RecieveMessage* request is set to 0
- **Long Polling** is when the RecieveMessage request queries all the servers for messages
 - Reduces empty responses by waiting until a message is in the queue before sending a response
 - Reduces false empty responses by querying all instead of a subset of servers
 - Returns messages as soon as they are available

AWS SQS Dead-letter Queue

- A queue that is used to store messages that cannot be processed for some reason
- Great for debugging applications and systems
- DLQs must be created as a standard of FIFO queue
 - Then configure as a DLQ
 - Can create via SQS Console, SDKs, CLI
 - Also can use AWS tools for PowerShell
- Uses a redrive policy which identifies the source queue, the dead letter queue, and any conditions that need to occur to redirect messages to the DLQ

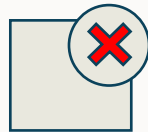


AWS SQS Dead-letter Queue



When to use

- With standard queues
 - Application doesn't depend on ordering
- Decrease number of messages
 - Poison-pill messages
 - Received but not processed



When not to use

- If program waits for dependent process
 - Active or available
- FIFO queue
 - Edit decision list (EDL)
 - Video editing suite

AWS SQS Best Practices

- Extended the message's **visibility timeout** to the maximum time it takes to process and delete a message
- Use the appropriate polling mode
- Configure a DLQ
- Batching can reduce costs
- Use message deduplication IDs to monitor duplicate sent messages

AWS Simple Notification Service (SNS)

- Used to send notifications from the AWS cloud
- Follows the “**pub-sub**” (**publish-subscribe**) model
 - Publishes communicate asynchronously and send messages to a topic (communication channel)
 - Subscribers consume messages when they are subscribed to the topic
- Messages delivered to clients using a push in an event-driven matter
 - EC2, S3, and RDS (Event sources), SQS, Lambda, HTTPS, Email, SMS (Event destinations)
 - Automate workflows and decouple infrastructure
- **Message filtering** can be used to only get notifications that are needed
- **Message fanout** is when a message is sent to a SNS topic then replicated and pushed to multiple endpoints for parallel processing

AWS SNS Delivery Retries

- SNS messages are processed and delivered immediately. If delivery is not successful, SNS implements a 4-phase retry policy:
 - Retries with no delay in between attempts
 - Retries with some minimum delay between attempts
 - Retries with some back-off model (linear or exponential)
 - Retries with some maximum delay between attempts



AWS Kinesis Data Streams

- Used to collect, process, and analyze real-time, streaming data
- **Data Producer:**
 - Application that sends the data records to a Kinesis data stream.
 - Assigned partition keys to records which determine what shard ingests the data record
 - **Sequence Number** is the unique identifier for each data record
- **Data Consumer:**
 - Distributed Kinesis application retrieving the most recent data from all shards in a stream
- **Data Stream**
 - Logical grouping of shards
 - No limits on number of shards in a stream



AWS Kinesis Data Streams

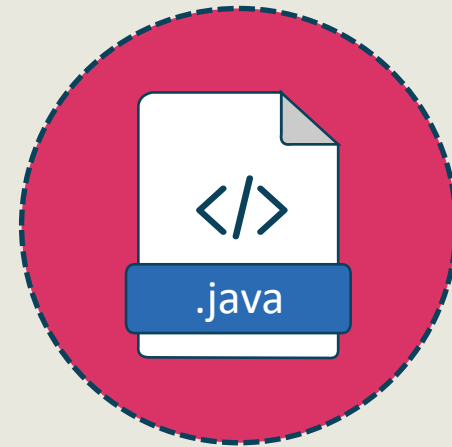
- **Shards:**
 - Base throughput unit of a Kinesis data stream
 - Contains ordered sequence of records
 - Add or remove shards as needed
 - 1 Shard = 1000 data records/s or 1MB/s
 - Enhanced Fan-out (Does not share consumers)
 - 1MB/s data input
 - 2MB/s data output
 - Non Enhanced Fan-out (Shared amongst all consumers)
 - 1MB/s data input
 - 2MB/s data output

AWS Kinesis Data Firehose

- Used to load streaming data into data stores and analytics tools from a Kinesis data stream, a Kinesis Agent, or the Kinesis Data Firehose API using the SDK
- Can also use CloudWatch Logs, Events and IoT as your data source
- Can invoke Lambda to transform incoming data and deliver it to a destination
- Fully managed; automatic scaling
- Capable of batching, compressing, and encrypting data before loading it
- Captures, transforms and loads the streaming data into S3, Redshift, Elasticsearch, and Splunk
- Customizable batch size and batch intervals

AWS Kinesis Data Analytics

- Quickly build SQL queries and Java applications using the built-in templates
- Completely **serverless**
- Automatic scaling to handle increases or decreases in volume
- Generate real-time alerts





AWS Kinesis Video Streams

- Managed service to stream live video from devices to the AWS cloud
- Can build applications for real-time video processing
- Connect and stream from millions of devices
- **Serverless**
- Enforces Transport Layer Security (TLS)-based encryption on data in flight
- Encrypts data at rest using AWS KMS

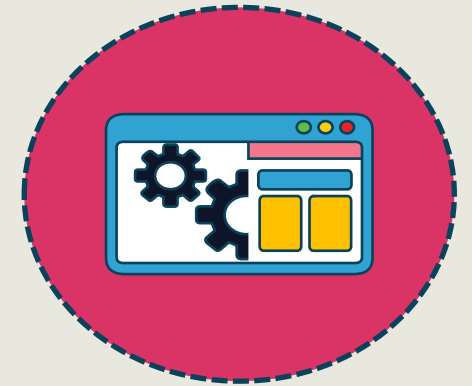


AWS Kinesis Video Streams Components

- **Producer:**
 - Source that sends data into a Kinesis video stream
- **Kinesis video stream**
 - Resource that transports live video, stores it and makes the data available
 - **Time-encoded data** is data in which records are in a time series
 - **Fragment** is a self-contained sequence of frames
 - Streams store incoming media data as **chunks**

AWS Step Functions

- **Serverless managed orchestration** for applications
- Use visual workflows to coordinate the components of distributed applications and microservices
- Based on the concepts of **tasks and state machines**
 - A task performs work by using an activity or a Lambda function, or by passing parameters via an API call
 - A state machine can express an algorithm as a number of states, their relationships, and inputs/outputs



AWS Serverless Application Model (SAM)

- Open-source framework for building serverless applications
- AWS Serverless Offerings:
 - Lambda, Lambda@Edge, AWS Fargate
 - S3, EFS
 - DynamoDB, Aurora, RDS Proxy
 - API Gateway
 - SNS, SQS, AppSync
 - AWS Step Functions
 - Kinesis, Athena

AWS Serverless Application Model (SAM)

- We need to create JSON or YAML configuration template
- When deployed, SAM transform and expands the SAM syntax into CloudFormation syntax.
- Declare all the same resources in a SAM template that you could a CF template
- Can build an application that use any runtime supported by AWS Lambda

AWS Serverless Application Model (SAM)

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetHtmlFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://sam-demo-bucket/todo_list.zip
      Handler: index.gethtml
      Runtime: nodejs6.10
      Policies: AmazonDynamoDBReadOnlyAccess
      Events:
        GetHtml:
          Type: Api
          Properties:
            Path: /{proxy+}
            Method: ANY

  ListTable:
    Type: AWS::Serverless::SimpleTable
```



AWS Lambda

- A serverless compute offering from AWS
- Executes code only when needed
- Automatic scaling
- Provision the amount of memory you want to allocate to your functions and Lambda allocates the proportioned CPU power, bandwidth and disk I/O
 - **Concurrent executions = (invocations per second) x (average execution duration in seconds)**
- Supports Node.js, Java, C#, Go, Python, Ruby and custom runtimes
- Supports synchronous and asynchronous invocation



AWS Lambda Components

- **Function**

- Program or script that Lambda runs. These functions are invoked by **events**. Events are processed and a response is returned
- **Function package** has all necessary components for your function Code, files, etc
- **Function handler** starts the actual code execution

- **Runtimes**

- Different languages can be used and still run in the same execution environment.
- Sits between Lambda and your code; relaying events, and responses between the two

- **Layers**

- ZIP archive that contains libraries, custom runtimes or other dependencies.
- Allows you to manage your dependencies separate from the unchanging code and resources



AWS Lambda Components

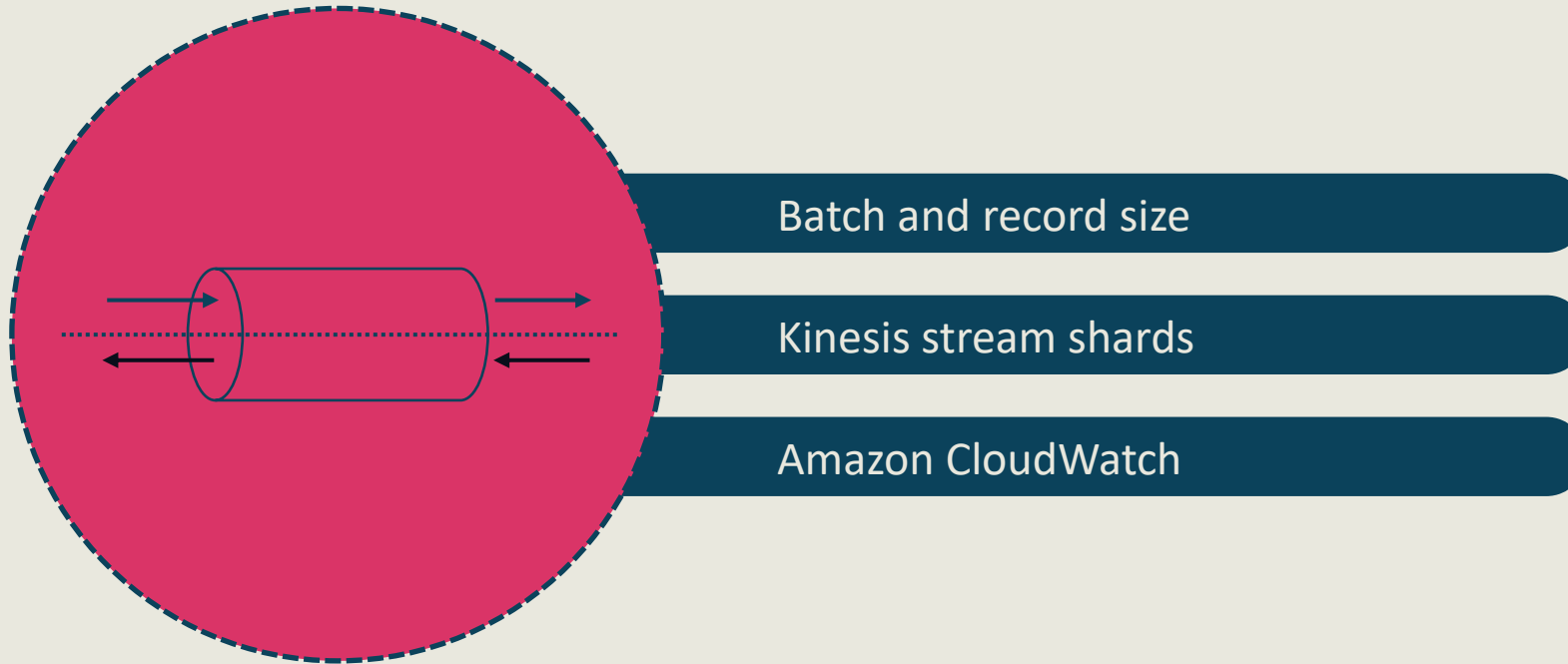
- **Event Source**

- AWS Server or custom service that triggers your Lambda function
- Can be **nonstreaming event sources**, also called the **push model**.
 - Amazon Echo, S3, SNS, and Cognito
- Can be **streaming event sources**, also called the **pull model**
 - Amazon Kinesis, or DynamoDB streams

- **Versions and Aliases**

- Secondary resources you can use to manage deployment and invocation
- Great for testing; each version has its own ARN; traffic shifting

AWS Lambda Function Streams



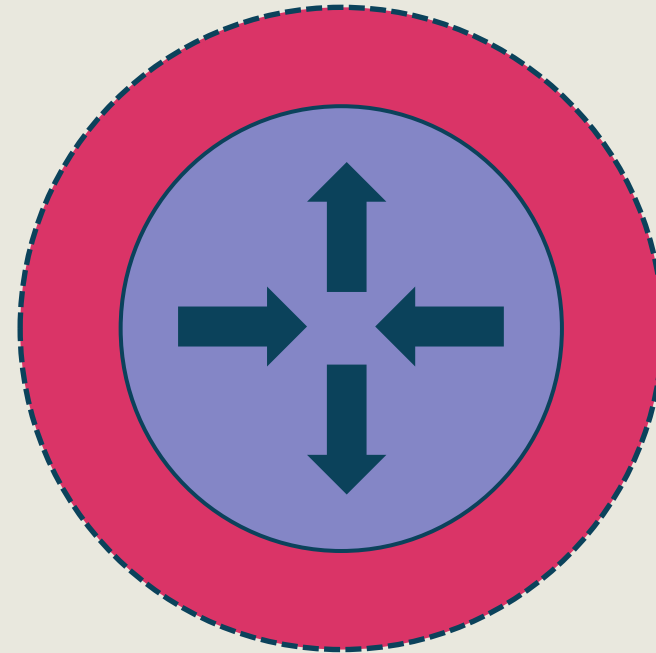


AWS Lambda@Edge

- Lambda that will run function for content that is being delivered by CloudFront
- Functions are executed closer to the viewer
- Functions will run in response to CloudFront events
- Managed service

AWS API Gateway

- AWS service for:
 - REST, HTTP and WebSocket APIs
 - Creates
 - Publishes
 - Maintains
 - Monitors
 - Secures





AWS API Gateway Concepts

- **API Deployment:** point-in-time snapshot of your API Gateway resources and methods; Must be deployed in order for clients to use
- **API endpoints:** *rest-api-id.execute-api.region.amazonaws.com*
- **API Key:** Used by an app developer who uses your API
- **API Stage:** Logical reference to a lifecycle state of your API
- **Private Integration:** Client accesses resources inside customer's VPC through private API endpoint. No travelling of the internet
- **Proxy Integration:** Can pass request and response to HTTP backend or send request as an input to Lambda



AWS API Gateway Integration

- **Method Requests**

- The public interface of your API; defines what your endpoint expects, required elements, etc
- Can specify requirements such as *Authorization* header here for AA

- **Integration Requests**

- Method request authenticated the request, we validated the structure, now we have to transform it
- You may want to remove something like *Authorization* header before forwarding; here is where we do that
- Not necessary for HTTP proxy or Lambda Proxy integration



AWS API Gateway Integration

- **Integration Response**

- Used to standardize the response so the method response can handle it
- Inverse of the integration request
- Use a regex pattern to identify the status code of your response

- **Method Response**

- Not necessary if using Lambda Proxy or HTTP proxy integration
- Much like method request; used to validate output to client
- Returns 200 OK by default
- You must create method responses before you can use a given status code in an integration response



AWS API Caching

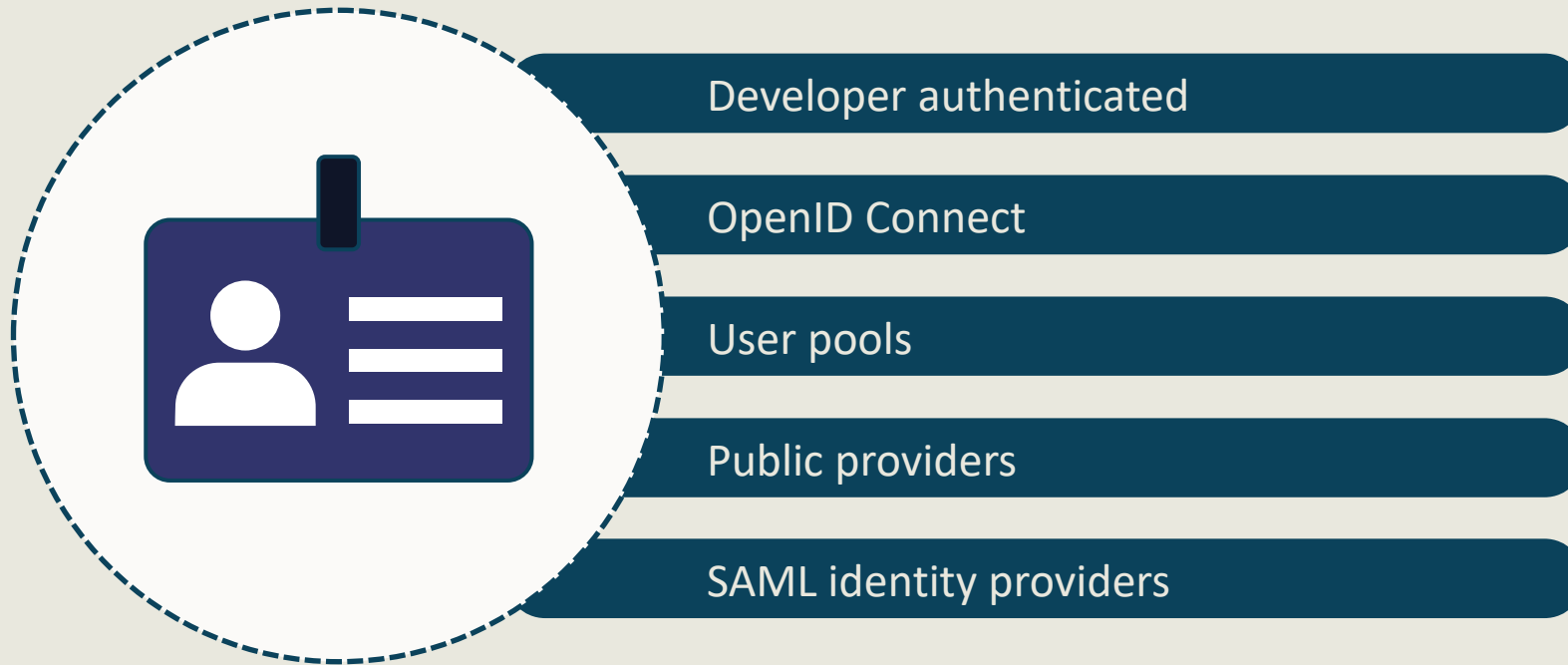
- Clients can invalidate an existing cache entry and reload it from the integration endpoint for each individual requests.
- Client must send request with **Cache-Control: max-age=0** header

AWS API Security – User Pools

- **COGNITO!**
- User Pools
 - User directories with sign-up and sign-in capabilities
 - Get JSON web token
 - Use as authorizer for existing API



AWS API Security – Identity Pools





AWS API CloudWatch Metrics

- API Gateway reports metrics that you can analyze and get insight on your API
 - **IntegrationLatency** measures the responsiveness of the backend
 - **Latency** measures the overall responsiveness of your API calls
 - **CacheHitCount** and **CacheMissCount** are used to optimize caching capabilities