# Domain 1: Monitoring, Logging, and Remediation

## 1.1 Implement metrics, alarms, and filters by using AWS monitoring and logging services

### Identify, collect, analyze, and export logs (for example, Amazon CloudWatch Logs, CloudWatch Logs Insights, AWS CloudTrail logs)

* + CloudWatch Logs
    - Monitor and store logs to help better understand and operate your systems and applications
    - Real-Time Application and System Monitoring
      * Monitor using log data
      * Can track number of errors that occur in your application logs and send you a notification whenever the rate of errors exceeds a threshold
    - Long-term log retention
      * Store log data indefinaitely in highly durable and cost effective storage without worrying about hard drives running out of space
      * Quickly move rotated and non-rotated logs files off host
  + Cloudwatch Logs Insights
    - Interactive, pay as you go and integrated log analytics capability for cloudwatch logs
    - Search and visualize logs
    - Allows you to understand your applications and make improvements and find and fix problems quickly
  + AWS CloudTrail Logs

### Collect metrics and logs using the CloudWatch agent

* + See Appendix

### Create CloudWatch alarms

### Create metric filters

### Create CloudWatch dashboards

### Configure notifications (for example, Amazon Simple Notification Service [Amazon SNS], Service Quotas, CloudWatch alarms, AWS Health events)

## 1.2 Remediate issues based on monitoring and availability metrics

### Troubleshoot or take corrective actions based on notifications and alarms

### Configure Amazon EventBridge rules to trigger actions

### Use AWS Systems Manager Automation documents to take action based on AWS Config rules

# Domain 2: Reliability and Business Continuity

## 2.1 Implement scalability and elasticity

### Create and maintain AWS Auto Scaling plans

* + See Appendix EC2 AutoScaling

### Implement caching

### Implement Amazon RDS replicas and Amazon Aurora Replicas

* + Read Replicas for Scalability
    - Up to 5 replicas
    - Same AZ, Cross AZ, Cross Region
    - Async replication
      * Reads eventually consistent
    - Replica can be promoted to its own database to accept reads
    - Application must update connection string to point to all replicas
    - Only for SELECT Statements
      * Writing to read replca can break the replication
    - Cost
      * Replica in same region not cost to transfer between AZ
      * Cross region replication pay for transfer
    - Read replicas can be setup as Mult AZ for DR
    - If the value for Max\_Allowed\_Packet parameter for a read replica is less than the Max\_allowed\_packet for the source DB instance, replica error can occur
      * Custom parameter
      * Specify the max size of data manipulation language (DML) that can be run on the DB
      * [Troubleshooting for Aurora - Amazon Aurora](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/CHAP_Troubleshooting.html#CHAP_Troubleshooting.MySQL.ReplicaLag)
  + Multi AZ
    - Sync replication
    - One DNS name for master and replicas
    - Auto failover to replicas if primary fail
    - No manual intervention
    - Not used for scaling. Replica is used for standby
    - Config
      * Zero downtime
      * Modify DB and enable multi AZ

### Implement loosely coupled architectures

### Differentiate between horizontal scaling and vertical scaling

* + Horizonal
    - Scale out
    - Add more instances
  + Vertical
    - Scale up
    - Increase size of instance

## 2.2 Implement high availability and resilient environments

### Configure Elastic Load Balancer and Amazon Route 53 health checks

* + See Appendix ELB

### Differentiate between the use of a single Availability Zone and Multi-AZ deployments (for example, Amazon EC2 Auto Scaling groups, Elastic Load Balancing, Amazon FSx, Amazon RDS)

* + EC2 Auto Scaling
    - See Appendix EC2 AutoScaling
  + Elastic Load Balancing
    - See Appendix ELB
  + Amazon FSx
  + Amazon RDS

### Implement fault-tolerant workloads (for example, Amazon Elastic File System [Amazon EFS], Elastic IP addresses)

### Implement Route 53 routing policies (for example, failover, weighted, latency based)

## 2.3 Implement backup and restore strategies

### Automate snapshots and backups based on use cases (for example, RDS snapshots, AWS

### Backup, RTO and RPO, Amazon Data Lifecycle Manager, retention policy)

### Restore databases (for example, point-in-time restore, promote read replica)

### Implement versioning and lifecycle rules

### Configure Amazon S3 Cross-Region Replication

### Execute disaster recovery procedures

# Domain 3: Deployment, Provisioning, and Automation

## 3.1 Provision and maintain cloud resources

### Create and manage AMIs (for example, EC2 Image Builder)

* + Overview
    - Amazon Machine Image
    - Customization of and EC2 Instance
      * You add your own software, config etc
      * Gold image
    - AMI are built for specific region
      * Can be copied across regions
    - You can launch EC2 from
      * Public AMI
        + AWS provide
      * Your own AMI
        + AMI you create
      * AWS Marketplace AMI
        + Ami created by third party
        + May need to purchase
  + No Reboot option
    - Enable you to create AMI without shutting down AMI Instance
    - By default not selected, instance will reboot when creating AMI
      * OS Buffers are not flushed in this mode
    - AWS Backup Plans to create AMI
      * Must use no-reboot option
  + EC2 Instance Migration between AZ
    - Uses AMI
    - Create AMI
    - Restore AMI in new instance in diff AZ
  + Cross account AMI sharing
    - You can share an AMI with another AWS account
    - Sharing does not affect ownership of AMI
    - Share if
      * AMI unencrypted or encrypted with customer managed key
    - If sharing encrypted volumes you must share customer managed key used to encrypt them
    - Cross Account AMI Copy
      * If you copy an AMI that has been shared , you become the owner of the new copy
      * The owner of the source AMI must grant read permissions for the storage that backs the AMI
      * If encrypted owner must share the keys
  + EC2 Image Builder
    - Automate creation of VM or container images
      * Automate creation, maintain, validate and test EC2 AMIs
    - Built AMI can be distributed to different regions
    - Can be run on a schedule
    - Free service
  + Can force users to only launch EC2 instances from pre-approved AMIs (AMIs tagged with specific Tags) using IAM Policies
  + Use AWS Config to find non compliant EC2 instances (instances started with non approved AMI)

### Create, manage, and troubleshoot AWS CloudFormation

* + See Appendix
  + Overview
    - Infrastructure as Code
    - Allows reproducing deployments
    - Declarative way
    - Can be version controlled
    - Templates
      * Uploaded to S3 and referenced in CloudFormation
      * Can’t edit previous template
  + Create
    - Resources created in correct order
    - Parameters defined in template
    - YAML and JSON
      * JSON sucks for CLoudformations
      * YAML Key value pairs
      * Nested objects
      * Arrays (-)
      * Multi line strings (|)
      * Comments
    - Resources
      * Mandatory
      * Diff AWS components
      * Can reference each other
      * AWS::aws-product-name::data-type-name
      * AWS Documentation has the syntax for the YAML
        + [AWS resource and property types reference - AWS CloudFormation (amazon.com)](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html)
      * Type
      * Properties
    - Parameters
      * Provide input to template
      * Reuse templates across company or regions / az
      * Controlled and can prevent errors thanks to types
      * When to use parameters
        + Is the resource config likely to change in the future
      * Settings
        + Type
        + Description
        + Contrainst
        + Etc
      * Reference a parameter
        + !Ref paramname
        + Can also reference other resources in template
        + Pseudo Parameters

Enabled by default

Built in parameters that reference AWS data

AWS::AccountID

AWS::NotificationARNs

AWS::NoValue

AWS::Region

AWS::StackId

AWS::StackName

* + - Mappings
      * Fixed variables within template
      * Hardcoded
      * When to use:
        + If you know in advance all the values that could be used
        + Safer control over template
      * Use mapping
        + Fn::FindInMap

!FindInMap [MapName, TopLevelKey, SecondLevelKey]

* + - Output
      * Optional
      * If exported then they can be imported into other stacks
      * View in console or cli
      * Cross stack collaboration allowing others to create stack
      * Cannot delete template that has output ref by another stack
      * To import
        + Fn::ImportValue function

!ImportValue

* + - Conditions
      * Used to control the creation of resources or outputs
      * Syntax:
        + LogicID: Function [ values ]
        + LogicID is created by you
        + Function

Fn::And

Fn::Equals

Fn::If

Fn::Not

Fn::Or

* + - Intrinsic Functions
      * Fn::Ref
        + Reference parameter or resource within the template
      * Fm::GetAtt
        + Get attribute of a resource
      * Fn::FindInMap
        + See mappings
      * Fn::ImportValue
        + See Output
      * Fn::join
        + Join values with a delimeter
        + Syntax

!Join [ delimeter, [ comman-delimited list of values ] ]

* + - * Fn::sub
        + Substitute
      * Conditionfunctions
        + See Conditions
  + Manage
    - Update
      * Can’t edit existing cloudformation template
      * Must upload / create new template
      * Change set preview
        + Shows what will change between new stack and already deployed stack
    - Delete
      * Delete stack and all resources deleted in correct order
      * All resources created by cloudformations deleted

### Provision resources across multiple AWS Regions and accounts (for example, AWS Resource Access Manager, CloudFormation StackSets, IAM cross-account roles)

* + AWS Resource Access Manager
    - Enables you to share specified AWS Resources that you own with other AWS accounts.
  + CloudFormation StackSets
    - Create update or delete stacks across multipe accounts and regions with single operation
    - Administrator account to create stacksets
    - Trusted accounts to create update delete stack instances from stacksets
    - When you update the stack set, all associated stack instances are updated throughout all accounts and regions
    - Ability to set max concurrent actions on targets (# or %)
    - Ability to set failure tolerance (# or %)
  + IAM Cross-Account Roles

### Select deployment scenarios and services (for example, blue/green, rolling, canary)

### Identify and remediate deployment issues (for example, service quotas, subnet sizing, CloudFormation and AWS OpsWorks errors, permissions)

* + Service Quotas
  + Subnet sizing
  + Cloudformation
  + Aws opsworks errors
  + permissions

## 3.2 Automate manual or repeatable processes

### Use AWS services (for example, OpsWorks, Systems Manager, CloudFormation) to automate deployment processes

* + OpsWork
    - See Appendix
  + Systems Manager
    - See Appendix
  + CloudFormation
    - See Appendix

### Implement automated patch management

* + See Appendix

### Schedule automated tasks by using AWS services (for example, EventBridge, AWS Config)

# Domain 4: Security and Compliance

## 4.1 Implement and manage security and compliance policies

### Implement IAM features (for example, password policies, MFA, roles, SAML, federated identity,resource policies, policy conditions)

* + Controls Internal Team accounts. NOT USER ACCOUNTS
  + Policy document bound to IAM User
    - Entity in AWS defines permissions for identity or resource
    - Allow statements
    - Deny statements
    - Policy types
      * Identity-based policy (IAM Policy)
        + Attach managed and inline policies to IAM identities

Users

Groups

roles

* + - * Resource-based policies
        + Attach inline polices to resources
        + S3 bucket policies and IAM Role Trust policies most common
      * AWS Organizations Service Control Policies (SCP)
        + Define max permissions for account members of an organization or OU
      * Access control Lists (ACL)
        + Control which principals in other accounts can access a resource to which the ACL is attached.
        + Similar to resource based policies
        + Does not use JSON structure
    - Policy Elements (JSON)
      * Effect
        + Allow or Deny
        + Implicit Deny
        + Most restrictive wins (Deny)
      * Principal
        + Account, user, role or federated user to which you want to allow or deny access
        + Only used with Resource-based policy
      * Action
        + List of actions the policy allows or denys
      * Resource
        + List of resources to which the action apply
        + Required for IAM Policy
        + Optional for Resource-based Policy
      * Condition
        + Circumstances under which the policy grants permission.
  + Authentication
    - Sign in to AWS
    - Users
      * Username and password
    - Federated users
      * Single Sign On
      * External Identity store (AD, Azure AD, LDAP, etc)
      * Assume IAM Role
    - Applications
      * Permissions to access AWS Resources can be assigned
      * Access/secret keys
        + Api level credentials
    - IAM Role
      * Policy document bount to IAM Role
  + Authorization
    - What you can do
    - Policy document
      * Explicitly grants permissions
        + Users

Can belong to multiple groups

* + - * + Groups

Collection of IAM Users

Cannot be nested

Not a principal

IAM document policy

* + - Default = New users have not permissions
  + MFA
    - Second factor auth
  + Access keys rotate
    - AWS Config can be used to identify access keys past due for rotation
  + Remove unnecessary credentials
    - Last time user logged in ( over x days) should probably be removed.

### Troubleshoot and audit access issues by using AWS services (for example, CloudTrail, IAM Access Analyzer, IAM policy simulator)

* + IAM Policy Simulator
    - Test policies

### Validate service control policies and permissions boundaries

### Review AWS Trusted Advisor security checks

### Validate AWS Region and service selections based on compliance requirements

### Implement secure multi-account strategies (for example, AWS Control Tower, AWS,Organizations)

## 4.2 Implement data and infrastructure protection strategies

### Enforce a data classification scheme

### Create, manage, and protect encryption keys

### Implement encryption at rest (for example, AWS Key Management Service [AWS KMS])

### Implement encryption in transit (for example, AWS Certificate Manager, VPN)

### Securely store secrets by using AWS services (for example, AWS Secrets Manager, Systems Manager Parameter Store)

### Review reports or findings (for example, AWS Security Hub, Amazon GuardDuty, AWS Config, Amazon Inspector)

# Domain 5: Networking and Content Delivery

## 5.1 Implement networking features and connectivity

### Configure a VPC (for example, subnets, route tables, network ACLs, security groups, NAT gateway, internet gateway)

* + See Appendix

### Configure private connectivity (for example, Systems Manager Session Manager, VPC endpoints, VPC peering, VPN)

* + See Appendix

### Configure AWS network protection services (for example, AWS WAF, AWS Shield)

## 5.2 Configure domains, DNS services, and content delivery

### Configure Route 53 hosted zones and records

### Implement Route 53 routing policies (for example, geolocation, geoproximity)

### Configure DNS (for example, Route 53 Resolver)

### Configure Amazon CloudFront and S3 origin access identity (OAI)

### Configure S3 static website hosting

## 5.3 Troubleshoot network connectivity issues

### Interpret VPC configurations (for example, subnets, route tables, network ACLs, security groups)

* + See Appendix

### Collect and interpret logs (for example, VPC Flow Logs, Elastic Load Balancer access logs, AWS WAF web ACL logs, CloudFront logs)

* + See VPC Flow Log Appendix
  + See Appendix ELB

### Identify and remediate CloudFront caching issues

### Troubleshoot hybrid and private connectivity issues

# Domain 6: Cost and Performance Optimization

## 6.1 Implement cost optimization strategies

### Implement cost allocation tags

### Identify and remediate underutilized or unused resources by using AWS services and tools (for example, Trusted Advisor, AWS Compute Optimizer, Cost Explorer)

### Configure AWS Budgets and billing alarms

### Assess resource usage patterns to qualify workloads for EC2 Spot Instances

* + See Appendix EC2

### Identify opportunities to use managed services (for example, Amazon RDS, AWS Fargate, EFS)

* + Fargate
    - Serverless EKS and ECS

## 6.2 Implement performance optimization strategies

### Recommend compute resources based on performance metrics

### Monitor Amazon EBS metrics and modify configuration to increase performance efficiency

* + See Appendix EBS

### Implement S3 performance features (for example, S3 Transfer Acceleration, multipart uploads)

### Monitor RDS metrics and modify the configuration to increase performance efficiency (for example, Performance Insights, RDS Proxy)

### Enable enhanced EC2 capabilities (for example, enhanced network adapter, instance store, placement groups)

* + See Appendix

## Appendix

Which key tools, technologies, and concepts might be covered on the exam?

The following is a non-exhaustive list of the tools and technologies that could appear on the exam. This list

is subject to change and is provided to help you understand the general scope of services, features, or

technologies on the exam. The general tools and technologies in this list appear in no particular order.

AWS services are grouped according to their primary functions. While some of these technologies will likely

be covered more than others on the exam, the order and placement of them in this list is no indication of

relative weight or importance:

AWS services and features

### Analytics:

* Amazon Elasticsearch Service (Amazon ES)
* Application Integration:
* Amazon EventBridge (Amazon CloudWatch Events)
  + Overview
    - Intercept events from AWS services
    - Intercept API call with cloud trail integration
    - Schedule event
    - Json payload for event passed to target
  + EventBridge Bus
    - Default bus = Cloudwatch Events (AWS services)
    - Partner Event BUS = events from SaaS services or applications (third parties)
    - Custom event BUS = events from own applications
    - Can be access from other accounts
    - Rules to process events
  + Can analyze events depending on schema
  + Schema registry
    - Allows you to generate code for your application that will know in advance how data is structured in the event bus
    - Schema can be versioned
* Amazon Simple Notification Service (Amazon SNS)
* Amazon Simple Queue Service (Amazon SQS)

### AWS Cost Management:

* AWS Cost and Usage Report
* AWS Cost Explorer
* Savings Plans

### Compute:

* AWS Application Auto Scaling
* Amazon EC2
  + AMI
  + Change instance
    - Only works with EBS Storage.
    - Stop instance
    - Change instance type
    - Restart instance
    - Note
      * EC2 Instance does not loose data
  + Enhanced Networking
    - SR-IOV
      * Higher bandwitch, higher PPS (packets per second), lower latency
      * Option I
        + Elastic Network Adapter
        + Up to 100 Gbps
      * Option 2
        + Intel 82599VF
        + Up to 10 Gbps – Legacy
      * Works for newer generation of EC2
    - Elastic Fabric Adapter
      * Improved for HPC, only works in linux
      * Greate for inter-node communications, tightly couples workloads
      * Leverages message Passing Interface (MPI)
      * Bypasses the underlying Linux OS to provide low-latency, reliable transport
  + EC2 Placement Group
    - Sometime you want control over the EC2 placement strategy
    - Strategy
      * Cluster
        + Instances grouped together in low same AZ, same rack, same hardware
        + Low-latency
        + High risk
        + Use

Big data jobs

Low latency network requirements

* + - * Spread
        + Instances spread across underlying hardware ( max 7 instances per group per AZ)
        + Minimize failure risk
        + Use

Application needs to be maximized high availability

Critical applications

* + - * Partition
        + Spread instances across diff partitions

Rely on different sets of racks within AZ

* + - * + 7 partitions per AZ
        + Scales to 100s of instances per group
        + use

Works good for Hadoop, Casssandre, Kafka

* + Shutdown and Termination Protection
    - Shutdown Behavior
      * Shutdown from within OS
        + Stop

Default

Ec2 instance will stop as well

* + - * + Terminate

Stops and deletes the Instance

* + - * + AWS CLI

InstanceInitiatedShutdownBehavior

* + - Termination Protection
      * Enabled = protect against accidental termination (deletion) in AWS Console or CLI
  + Troubleshooting EC2 Launch
    - InstanceLimitExceeded
      * reached your limit of max number vCPUs per region
      * on-Demand and Spot instance limits are set per region
      * Resolution
        + Launch instance in diff region or request AWS to increase your limit
    - InsufficientInstanceCapacity
      * AWS does not have enough On-Demand capacity in the AZ
      * Resolution
        + Wait for capacity to become available
        + If requesting more than one instance, reduce the number requested

Possibly request each instance one by one instead of all at once.

* + - * + Request diff instance type
        + Launch in diff AZ
    - InstanceTerminatesImmediately
      * Pending to terminate
      * Reached your EBS volume limit
      * EBS snapshot is corrupt
      * Root EBS volume is encrypted and you do not have permissions to access the KMS key for decryptions
      * The instance store-backed AMI used is missing a required part
      * Examin EC2 console
        + Description tab will tell you why
    - SSH
      * Unprotected Private Key file
        + Incorrect permission on private key (pem file)
      * Host key not found, Permission denied, or Connection closed
        + Incorrect username
      * Connection Timeout
        + Security Group not configured correctly
        + NACL not configured correctly
        + Route table incorrect
        + Instance does not have IPv4
        + CPU load on instance is high
  + EC2 Launch Types
    - On-Demand
      * Short workloads
      * Predictable pricing
      * Highest cost but no upfront payment
      * No long-term commitment
    - Reserved
      * Minimum 1 year or 3 years
      * Up to 75% discount
      * Reserved Instances
        + Long workloads
      * Convertible Reserved workloads
        + Long workloads with flexible instances
        + Can change the EC2 instance type over time
        + Less discount
      * No upfront , partial upfront, all upfront payment options
        + More upfront = greater discount
      * Scheduled-Reserved Instanaces
        + Deprecated
        + Launch within time window you reserve
        + When you require a fraction of day/week/month
    - Spot Instances
      * Up to 90% discount
      * Short workloads or resilient workloads that don’t loose what they have already done.
      * Cheapest
      * Can lose instances if someone pays more for it ( spot price changes over time )
      * No critical jobs or databases
      * Spot Block
        + Block, prevent, AWS from reclaiming your instance for 1 to 6 hours
        + Deprecated
      * One time request type
        + Will not launch again after it has met its price once.
      * Persistent request type
        + Will launch everytime the spot price is below your max
      * Terminate spot instance
        + Cancel request
        + Terminate spot instances running
      * Spot Fleet
        + Set of spot instances and on-demand instances
        + Will try to meet target capacity with price constraints
        + Strategies

Lowest price

Lauch instances from pools with lowest price first

Good for short workloads

Diversified

Distributed across all pools

Great for availability and long workloads

Capacity optimized

Pool with the optimal capacity of the number of instances

* + - Dedicated Hosts
      * Entire physical server, your software/os on dedicated server
      * Control instance placement
      * Use if you have compliance requirements
      * Allow you to use existing server-bound software licenses
      * 3 year commitment
      * More expensive
    - EC2 Dedicated Instances
      * Instances running on hardware that’s dedicated to you, VMs running on host dedicated to you.
      * May share hardware with other instances in your account
      * No control over instance placement
  + Burstable Instances
    - Instance can burst CPU for unexpected processing
    - Burst Credits
      * Burst credits are used during burst periods
      * If low credits then you probably are using the wrong instance type
      * If out of credits then cpu will drop way down
    - Unlimited instances do not use burst credits
      * Cost more when bursting
  + Elastic IP
    - Fixed IP like Static IP
    - When an instance starts it changes its IP (usually not the same all the time). Like DHCP
    - Attached to one instance at a time
    - Do not pay if attached to server
    - Can only have 5 per account
      * Should avoid using elastic IP
  + Cloudwatch for EC2
    - See Cloudwatch
  + Status Checks
    - Automatic checks to ID hardware and software issues
    - System Status Checks
      * Monitors problems with AWS systems
      * Issues on the physical host
        + Loss of network connectivity
        + Loss of system power
        + Software issues on the physical host
        + Hardware issues on the physical host that impact network reachability
      * Personal Health Dashboard
        + To check if there is any scheduled critical maintenance by AWS
      * If any problems you will need to stop and restart the EC2 instance
        + This migrates the instance to another host
    - Instance status check
      * Monitors software / network config of your instance
        + Incorrect networking or startup configuration
        + Exhausted memory
        + Corrupted file system
        + Incompatible kernel
      * Reboot the instance or change instance config
    - Automate
      * Cloudwatch Metrics and Recoery
        + Monitor

SttusCheckFailed\_System

StatusCheckFailed\_Instance

StatusCheckFailed (both)

* + - * + Cloudwatch Alarm

Monitor and recover instance with same private/public IP etc

Send notification using SNS

* + - * + AutoScaling Group

Set min/max/desired to 1

If issue then new instance will be started

* + EC2 Hibernate
    - In memory RAM is preserved
    - Like pause
    - OS is not stopped / restarted
    - Under the Hood
      * RAM must be written to the EBS root volume
      * Volume must be encrypted
    - Use cases ( when it makes sense to hibernate as opposed to restart )
      * Long running processing
      * Saving the RAM state
      * Services that take time to initialize
    - Only supports the following instances
      * C3,C4,C5,M3,M4,M5,R3,R4,R5
    - Instance RAM must be less than 150 GB
    - Not supported for bare metal
    - AMI: Linux 2, Linux AMI, Ubuntu and Windows
    - Root Volume must be EBS and encrypted
    - Only avail for on Demand and reserved instance
    - Cannot be hibernated more than 60 days
* Amazon EC2 Auto Scaling
  + Overview
    - Scale out to match an increased load
    - Scale in to match decreased load
    - Ensure we have a minimum and maximum number of machines running
    - Automatically register / deregister instances to a load balancer
    - Attributes
      * Launch configuration
        + AMI+instance type
        + EC2 User Data
        + EBS Volumes
        + Security Groups
        + SSH Key Pair
      * Min Size / Max Size / Initial Capacity
      * Network + Subnets information
      * Load balancer information
      * Scaling policies
    - Auto Scaling Alarms
      * Possible to scale an ASG based on Cloudwatch Alarms
      * Alarm can monitor a metric
      * Metrics are computed for the overall ASG instances
      * Based on alarms
        + Scale-out policies
        + Scale-in Policies
    - Auto Scaling Rules
      * Target Average CPU Usage (average over all instances)
      * Number of requests on the ELB per instance
      * Average Network in
      * Average Network out
      * Custom Metric
    - Brain Dump
      * Scaling policies can be on CPU, Networ, etc and can even be on custom metrics or based on schedule
      * Can use launch configs or Launch templates
      * Update ASG means to provide new Launch config / template
      * IAM roles attached to ASG will get assigned to EC2
      * ASG is free pay for only what is launched
      * If instance gets terminated, then ASG will create new instance automatically
      * ASG can terminate instances marked as unhealthy by a LB and then replace them
    - Available for scalable AWS Resources
      * EC2 ASG
        + Launch or terminate EC2 instances
      * EC2 Spot Fleet requests
        + Launch or terminate instances from spot fleet request
        + Automatically replace instances that get interrupted for price or capacity reasons
      * ECS
        + Adjust ECS service desired count up or down
      * DynamoDB (table or global secondary index)
        + Adjust write capacity and read capacity
      * Aurora
        + Dynamic Read Replicas
  + Scaling Policies
    - Dynamic Scaling Policies
      * Target Tracking Scaling
        + Most simple
        + Adjust capacity depending on target metic
        + Optimize for availability = 40%
        + Balance availabity and cost = 50%
        + Optimize for cost = 70%
        + Can choose own metric and target value
        + Can set

Disable scale in

Cooldown period

Warmup time

* + - * Simple / Step Scaling
        + When CloudWatch alarm is triggered then add / remove x units
      * Scheduled actions
        + Anticipate scaling based on known usage patterns
    - Predictive Scaling
      * Continuously forecast load and schedule scaling ahead
    - Metrics to scale on
      * CPUUtilization
        + Average CPU utilization across instances
      * RequestCountPerTarget
        + Number of requests per EC2 instance
      * Average Network In / Out
      * Any custom metric
    - Scaling cooldown
      * Period after scaling activity (in or out)
      * Default 300 seconds
      * During cooldown ASG will not launch or terminate additional instances
      * Allows for metrics to stabilize
      * Suggest using ready to use AMI so they start / stop faster
  + Lifecycle Hooks
    - By default an instance goes into service immediately
    - Hooks allow you to add extra steps before the instance goes into service or terminates
    - Pending State
      * You can perform steps between pending and inService states
    - Terminate State
      * Perform action before system terminates
    - Use Cases:
      * Cleanup
      * Log extraction
      * Special health checks
    - Can integrate with EventBridge, SNS and SQS
  + Launch Configuration vs Launch Templates
    - Both
      * Can specify
        + AMI
        + Instance Type
        + Key pair
        + Security groups
        + Etc
      * Can’t edit
    - Launch Config
      * Depricated
      * Must be re-created every time
    - Launch Template
      * Versioning
      * Parameter subset
        + Partial config for reuse and inheritance
      * Provision using both On-Demand and Spot or mixed
      * Supports
        + Placement Groups
        + Capacity Reservations
        + Dedicated hosts
        + Multiple instance types
        + T2 unlimited burst feature
    - SQS with ASG
      * Scale ASG based on messages in SQS
      * Create cloudwatch metric / alarm on queue length
    - ASG Healthchecks
      * High availability = 2 instances in at least 2 AZ ( multi-AZ ASG )
      * Types
        + EC2 status Checks
        + ELB Health Checks
        + Custom health checks

Send health via CLI or SDK

* + - * ASG will Terminate and relaunch new instance if one becomes unhealthy
      * CLI
        + Set-Instance-Health

Use with custom health checks

* + - * + Terminate-instance-in-auto-scaling-group
    - Troubleshooting
      * Cant launch new instances when other instances are already running
        + ASG group has reached the limit set by the Max Capacity
        + Need to increase this value
        + Capacity issue in AZ
      * Launching EC2 instances is failing
        + Security group does not exist
        + Key pair does not exist
      * If there are issue launching EC2 instances over 24 hours, ASG will suspend scaling processes (administration suspension
  + CloudWatch for ASG
    - Metrics are collected every 1 min
    - ASG Metrics (must enable at ASG to see metrics)
      * GroupMinSize
      * GroupMaxSize
      * GroupDesiredCapacity
      * GroupInServiceInstances
      * GropuPendingINstances
      * GroupStandbyInstances
      * GroupTerminatingInstances
      * GroupTotalInstances
    - EC2-Level Metrics ( enabled by default)
      * CPUUtilization, etc
      * Basic = 5 minute
      * Detailed = 1 minute
* Amazon EC2 Image Builder
* AWS Lambda

### Database:

* Amazon Aurora
* Amazon ElastiCache
* Amazon RDS
  + Overview
    - Relational Database Service
    - Managed SQL language DB
    - Types
      * Postgres
      * MySQL
      * MariaDB
      * Oracle
      * Microsoft SQL Server
      * Aurora
    - Managed by AWS
      * Automated provisioning
      * Continuous backups
        + Automatically enabled

Daily full

Transaction every 5 minutes

Point in time restore within 5 minutes

* + - * + 7 day retention (can be up to 35)
        + DB Snapshots

Manually triggered by user

Retain as long as you want

* + - * Monitoring
      * Read replicas
      * Multi az
      * Maintenance windows
      * Scaling
        + Auto scaling

Scales when storage gets close to threshold

Max storage threshold. (don’t grow over this)

* + - * Storage backed by EBS

Management, Monitoring, and Governance:

* AWS CloudFormation
  + User Data
    - User scripts
    - Pass entire script using Fn::Base54
    - User data script log is in /var/log/cloud-init-output.log
  + Cfn-init
    - CloudFormation Init
    - Stack must include AWS::CloudFormation::Init resource metadata
    - Makes complex scripts readable
    - Logs = /var/log/cfn-init.log
    - More declarative
  + Cfn-signal
    - Wait conditions
      * Wait for template to wait until it gets signal
      * Can be used to detect fail or success
      * Works with cfn-init
      * Wait condition in metadata
    - Failures troubleshooting
      * Ensure the AMI has the AWS Cloudformation Helper scripts installed
      * Verify cfn-init and cfn-signal command run successfully by checking logs
        + Must disable rollback to view logs
      * Must have connection to internet
        + This is how it talks to cloud formation service
        + Curl -l <https://www.amazon.com> to test internet connection
    - Creation Policy (ASG only plus a few others)
      * Waits for signal of x number to show the ASG has been created
    - Update Policy (ASG Only plus a few others)
      * Attributes
        + AutoScalingReplacingUpdates

Creates new ASG with new config and terminate old one

* + - * + AutoScalingRollingUpdates

Specifies how many instances remain online and how many will be updated at one time.

* + - * + AutoScalingScheduledActions

Allows you to ignore ASG actions that would grow or shrink the number of systems running

* + Rollback
    - Stack creation fails it will rollback (all deleted)
    - Can disable rollback
    - If updating stack and it fails then the stack will rollback to previous know working stack
      * Here you will be able to see what happened it the logs
    - Update\_rollback\_failed
      * Can’t rollback
      * Fix error manually
      * Or skip resources marked as rollback
  + Nested Stacks
    - Allow you to isolate repeated patterns /common components in separate stacks and call them from other stacks
  + ChangeSets
    - Won’t say if it will be successful but they show what will change during an update
    - Creating change set can update stack
  + Drift
    - Manual changes can be made to stack
    - Drift detects any changes to the stack not made by cloudformation
  + Deletion Policy
    - Controls what happens when the cloudformation template is deleted
    - DeletionPolicy
      * Retain
        + Preserve / backup in case of cloudformation deletes
      * Snapshots
        + EBS vol, ElastiCache cluster, elasticache replication group, RDS DBInstance, RDS DBClustr, Redshift cluster
        + Instance deleted but snapshot made before deletion
      * Delete
        + Default behavior
        + (except for AWS:RDS::DBCluster default is Snapshot)
        + To delete S3 bucket all objects need to be deleted from bucket first
  + Termination protection
    - Prevents accidental deletion
  + DependsOn
    - Resource will not be created until the dependson resource has been created.
    - Allows one resource to be completed first
    - Orders resource creation
  + Stack Policies
    - JSON file
    - Similar to IAM policy
    - Can prevent resources from being updated accidentally
    - Can overrule policy during update
* AWS CloudTrail
* Amazon CloudWatch
  + OverView
    - Monitoring service for AWS cloud resources and the applications you run on AWS
    - Collects and tracks metrics
    - Collect and monitor log files
    - Set alarms
    - Basic metrics collected every 5 minutes (free)
    - Detailed metrics collected every 1 minute (paid)
    - Retention
  + Metrics are provided automatically for a number of AWS products and Services
    - EC2 instances
      * CPU
        + CPU Utilization
        + Credit Usage/Balance (burst)
      * Network
        + Network In / Out
      * Disk
        + Read / Write for Ops / Bytes ( only for instance store )
      * Status Checks
        + Is the instance healthy or not
        + Instance status = check EC2VM health
        + System Status = Check the underlying hardware
      * RAM is not inclued
      * Custom Metrics
        + Basic every 1 minute
        + Hi resolution every 1 second
        + RAM
        + Application Level Metrics
        + Instance custom metrics requires an IAM Role to push to Cloudwatch
        + Requires cloudwatch agent
    - EBS Volumes
    - Elastic Load Balancer (ELB)
    - Auto Scaling
    - EMR job flow
    - RDS DB instances
    - DynamoDB tables
    - ElastiCache Cluster
    - Redshift clusters
    - OpsWorks stacks
    - Route53 health checks
    - SNS topics
    - SQS Queues
    - SQF workflows
    - Storage gateways
  + Unified Cloudwatch Agent
    - Overview
      * For virtual servers
        + EC2 Instances
        + On Prem Servers
        + Etc
      * Collect additional system-level metrics
        + RAM
        + Processes
        + Used disk space
        + Etc
      * Collect logs to send to cloudwatch logs
        + Default EC2 instance does not send logs
      * Centeralized configuration of agent using SSM Parameter Store
      * IAM Permissions need to be correct to push to Cloudwatch from the agent
      * Default namespace for metics collected by agent start with CWAgent
        + Configurable
      * Procstat Plugin
        + Collect metrics and monitor system utilization of individual processes
        + Supports linux and windows
        + Select which processes to monitor

Pid\_file: name of process identification number files they create

Exe: process name that match string you specify (regex)

Pattern: Command lines used to start he process (regex)

* + - * + Metrics collected by procstat plugin begin with procstat
* AWS Command Line Interface (AWS CLI)
* AWS Compute Optimizer
* AWS Config
* AWS Control Tower
* AWS License Manager
* AWS Management Console
* AWS OpsWorks
  + Overview
    - Use Chef and Puppet to perform server config automatically or repetitive actions
    - Work with EC2 or onPrem
    - Managed Check and Puppet
    - Alternative to SSM
* AWS Organizations
* AWS Personal Health Dashboard
* AWS Secrets Manager
* AWS Service Catalog
* AWS Systems Manager
  + OverView
    - Helps you manage EC2 and on PRemisses systems at scale
    - Get operational insights about the state of infrastructure
    - Easily detect problems
    - Patching automation for enhanced compliance
    - Linux and Windows
    - Integrated with CloudWatch metrics / dashboards
    - Integrated with AWS Config
    - Free Service
    - SSM agent must be installed on system we will control
    - Installed on Linux 2 AMI by default
    - Can push to EC2 Instance
    - EC2 instanc needs IAM roles to allow SSM actions
  + Tags and Resource Groups
    - Tags can be used for resource grouping, etc
    - Resource groups allow you to work on multiple resources at one time
  + Documents
    - JSON or YAML
    - Define parameters
    - Define actions
    - Used to
      * Run command
      * State manager
      * Patch manager
      * Automation
      * Retrieve info from parameter store
  + Run Command
    - Execute a document
    - Run command across multiple instances (using resource groups)
    - Rate control ( how many instances to run against
    - Error Control
    - Integrated with IAM and CloudTrail
    - No need for SSH
    - Command output can be shown in
      * Console
      * Sent to S3 bucket
      * Cloudwatch logs
    - Send notifications to SNS about status
    - Can be invoked using EventBridge
  + SSM Automation
    - Simplifies common maintenance and deployment tasks of EC2 instances and other resources
    - Automation runbook
      * SSM Document of type automation
      * Defines actions performed on your EC2 instance or resource
      * Predefined runbook or custom runbooks
    - Triggered
      * Manually by AWS Console, AWS CLI or SKD
      * EventBridge
      * Schedule using maintenance windows
      * SQS Config rule remediation
  + SSM Inventory
    - Collect meta data from managed instances ( including on Prem)
    - Metadata includes
      * Installed software
      * OS Drivers
      * Config
      * Isntalled updates
      * Running service
      * Etc
    - Can be viewed in AWS Console or stored in S3 and query and analyze using Athena and QuickSight
    - Can spcify metadata collection interval
    - Query data from multiple AWS Accounts and regions
    - Create custom Inventory for custom metadata
  + State Manager
    - Used to automate process of keeping your managed instances in a state that you define
    - Use cases:
      * Bootstrap instances with software
      * Patch OS/Software update on schedule
      * Etc
    - State Manager Association
      * Defines the state that you want to maintain
      * Specify schedule when this will be applied
    - Use SSM Documents to create an association
  + Patch Manager
    - Automate process of patching instances
    - Both AWS and on prem
    - Linux and windows
    - On demand or on schedule
    - Scan instances and generate patch compliance report (missing patches)
      * Which can be sent to S3
    - Patch Baseline
      * Which patches should / should not be installed
      * Ability to create custom patch baselines ( specifying approved/rejected patches)
      * Patches can be autoapproved
      * By default install only critical patchs and security patches
    - Patch Groups
      * Associate set of instances with patch baseline
      * Instances should be defined with the tag key Patch Group
      * An instance can only be in one Patch Group\
      * Patch group can only be registered with one patch baseline
  + SSM Maintenance Window
    - Defines a schedule for when to perform actions on your instances
    - Contains
      * Schedule
      * Duration
      * Set of registered instances
      * Set of registered tasks
  + Session Manager
    - Allows you to start secure shell ( SSH ) on your EC2 instances and on prem servers
    - Access it through AWS Console, AWS CLI, or Sesseion manager SDK
    - Do not need SSH access, bation host or ssh keys
    - Requires
      * SSM agent needs to be installed
      * IAM Permissions
        + Control with user/groups can access Session Manager and which instances
        + Use tags to restric access to only specific EC2 instances
        + Access to SSM + write to S3 + write to cloud watch
        + Restrict commands a user can run in a session
    - Supports linude , MacOS and Windows
    - Log connections to your instances and Executed command ( regular SSH does not do this)
      * Can be sent to S3 or Cloudwatch logs
    - CloudTrail can intercept startSession events
* AWS Systems Manager Parameter Store
  + Securly store config and secrets
  + Optional seamless encryption using KMS
  + Serverless, scalable, durable, easy SDK
  + Version tracking of config / secrets
  + Config management using path and IAM
  + Notifications with CloudWatch Events
  + Integration with CloudFormation
  + Hierarchy
    - Custom created
  + Reference Secrets from secret manager
  + Reference AWS parameters
  + Parameter store tiers
    - Standard
      * Free
      * 10000
    - Advanced
      * .05 per parameter per month
      * 100000 parameters
  + Parameter Policies
    - Only for Advanced Tier
    - Allow to assign TTL to parameter to force updating or deleting sensitive data
    - Can assign multiple polices at one time
* AWS tools and SDKs
* AWS Trusted Advisor

### Migration and Transfer:

* AWS DataSync
  + Replicate all file attributes and metadata
* AWS Transfer Family

Networking and Content Delivery:

* AWS Client VPN
* Amazon CloudFront
* Elastic Load Balancing
  + Scalability
    - Virtical
      * Increase size of your instance
      * Scale up
    - Horizontal (elasticity)
      * Increase the number of instances
      * Scale out
      * Distributed systems
      * Auto scaling group
      * Load balancer
  + High availability
    - At least two AZ
    - Goal to survive AZ loss
    - Linked to Horizontal scaling
    - Auto scaling group multi AZ
    - Load balancer AZ
  + Over View
    - Forward traffic to multiple backend servers (downstream)
    - Only expose single point of access to app (Single DNS )
    - Seamlessly handle failures of downstream instances
    - Regular health checks
      * Knows if resource is properly working
      * ELB disables traffic to node if the health is bad
      * Uses port and route to check health
    - SSL Termination
    - Stickiness with cookies
    - High availability across zones
    - Separate public traffic from private traffic
    - Managed load balancer (
      * AWS guarantees it is working
      * Takes care of upgrades, maintenance high availability
      * Provies only few configuration settings
    - Costs less than setting up your own
    - Integrated with many AWS services
  + Types of load balancers
    - Classic
      * V1 – old generation 2009
      * http, https (Layer 7), tcp (Layer 4), ssl
      * do not use (deprecated)
      * Health checks are TCP or HTTP based
      * Fixed hostname
        + Xxxx.region.elb.amazonaws.com
      * OutofService
        + Health check failed
        + EC2 instance not registered with CLB yet
    - Application load balancer
      * 2016
      * http https (Layer 7), websock
      * mutlple HTTP applications
      * multiple apps on same machine
      * Target Group
        + EC2 instances

Can be autoscale group

* + - * + ECS tasks
        + Lambda functions
        + IP Address
        + Health check done at target group level
      * Route Routing
        + Routing tables to different target groups
        + Based on path in URL (example.com/users)
        + Based on hostname in URL (users.example.com)
        + Routing based on query string, headers
      * ALB good fit for micro services and container-base apps
      * Port mapping feature
        + Allows redirect to dynamic port in ECS (elastic Container Service)
      * Fixed hostname
        + Xxxx.region.elb.amazonaws.com
      * Application Serves do not see IP of client directly
        + Tru IP of client in X-Forwarded-For
        + Can also get Port (X-Forwarded\_Port) and Proto (X-Forwarded-Proto)
    - Network Load Balancer
      * 2017
      * TCP, tls, udp (Layer 4)
      * Handle millions of request per second
      * Less latency – for extreme performance
      * One static IP per AZ
      * Support assigning Elastic IP
      * Not in the AWS free tier
      * Target groups
        + EC2 instances
        + IP Address
        + Application Load Balancer
        + Lambda
      * Server Security Group needs to allow traffic from anywhere not just the NLB security group because the NLB passes traffic through
    - Gateway Load Balancer
      * 2020
      * Operates at lay 3(Network) IP Protocol
      * Deploy scale and manage fleet of 3rd party network virtual appliances ins AWS
        + Firewalls, intrusion detection and preventions systems, deep packet inspection systems…
      * CAN BE used to analyze traffic
      * Combines the following functions
        + Transparent Network Gateway

Single entry/exit for all traffic

* + - * + Load Balancer

Distributes traffic to your virtual appliances

* + - * Uses GENEVE protocol on port 6081
      * Target groups
        + EC2 instances
        + IP Addresses
  + Security Groups for LB
    - Users are allowed to Load balancer
    - Resource security group should only allow security group traffic from ELB
  + Sticky Sessions
    - Session Affinity
      * Client is always redirected to the same instance behind LB
    - Classic load balancer and Application load balancer
    - Cookies used with expiration date for stickiness
    - Use case:
      * Make sure user does not loose session data
    - Could cause imbalance
    - Cookies
      * Application bases cookie
        + Custom cookie

Generated by application

Each target group gets a cookie name

Can’t use

AWSALB, AWSALBAPP,AWSALBTG

* + - * + Application Cookie

Generated by the LB

Cookie name = AWSALBAPP

* + - * Duration based Cookie
        + Generated by LB
        + Name = AWSALB or AWSELB (clb)
  + Cross Zone Load Balancing
    - The load balancer in each AZ will distribute traffic evenly amongst all EC2 instances in all AZ
    - ALB this is always on and can’t disable
      * Not charge for inter AZ data
    - NLB disabled by default
      * You pay for inter AZ data transfer
    - CLB disabled by default
      * No charge for inter AZ
  + SSL Certificates
    - LB does HTTP SSL Termination
    - Certificates are managed by AWA Certificate Manager
    - Server Name Indication
      * Solves the problem of loading multiple SSL certificates onto one webserver
      * New protocol requires the client to indicate the hostname of the target server in the initial ssl handshake
      * Only works for ALB and NLB, or CloudFront
  + Connection Draining
    - CLB – Connection Draining
    - ALB or NBL – Deregistration Delay
    - Before stopping an instance, the LB gives some time for existing requests to complete
    - 1 to 3600 seconds
    - Default 300
    - 0 disables
  + Health Checks
    - Statuses
      * Initial
        + Registering the target
      * Health
      * Unhealthy
      * Unused
        + Target is not registered
      * Draining
        + Target is de-registering
      * Unavailable
        + Health checks disabled
    - If target group contains only unhealthy targets, ELB routes requests across all targets
  + Monitoring
    - Metrics push to Cloudwatch from all LB
      * BackendConnectionErrors
      * HealthyHostCount/UnHealthyHostCount
      * HTTPCode-Bakckend
        + 2xx
        + 3xx
        + 4xx
        + 5xx
      * Latency
      * RequestCount
      * RequestCountPerTarget
      * SergeQueueLength
      * SpillOverCount
        + Request rejected because the queue was too long
  + Troubleshooting
    - Successful request = 200
    - Unsuccessful at client = 4xx
      * 403 = client sent malformed request that does not meet HTTP Specifications
    - Unsuccessful at server = 5xx
      * 503 = Service Unavaiable
      * 504 = Gateway Timeout
  + Logging
    - Access logs from LB can be store in S3
    - Contain
      * Time
      * Client IP
      * Latencies
      * Request Path
      * Server Response
      * Trace ID
    - Helpful for compliance
    - Already encrypted
  + Tracing
    - Request tracing – each HTTP request has an added custom header added
      * X-Amzn-Trace-ID
    - Help track single request
    - ALB not working with X-Ray
  + Target Group Attributes
    - Detregistration\_delay.timeout\_seconds
      * Time the LB waits before deregistering a target
    - Slow\_start.duration\_seconds
      * Default target receives full share of requests once it is registered
      * Gives healthy target time to warmup before the load balancer sends them a full request
      * Exit slow start when:
        + Duration period ends
        + Target unhealthy
    - Load\_balancing.algorithm.type
      * How the load balancer selects targets when routing requests
        + Round Robin

Equally choose the targets from group

ALB and CLB

* + - * + Least Outstanding Requests

Next instanc to receive the request is the instance with the lowest number of pending/unfinished requests

Works with ALB and CLB

* + - * + Flow Hash

NLB

Target based on protocol, source/destination IP, source /destination port and TCP sequence number

Each TCP/UDP connection is routed to a single target for the life of the connection (suedo sticky)

* + - Stickiness.enabled
    - Stickiness.type
      * Application-based or duration-based
    - Stickiness.app\_cookie.cookie\_name
      * Name of the application cookie
    - Stickiness.app\_cookie.duration\_seconds
      * Application-based cookie expiration period
    - Stickiness.lb\_cookie.duration\_seconds
      * Duration-based cookie expiration perios
  + ALB Rules
    - Default rule last rule
    - Process in order
    - Supported actions
      * Forwarded
      * Redirect
      * Fixed response
    - Rule conditions
      * Host-header
      * HTTP-requested method
      * Path-pattern
      * Source-ip
      * http-header
      * query-string
    - target group weighting
      * specify weight fro each target group on a single rule
      * blue / green deployment
* AWS Firewall Manager
* AWS Global Accelerator
* Amazon Route 53
* Amazon Route 53 Resolver
* AWS Transit Gateway
  + Transit Gateway
    - Transitive peering between thousands of VPC and on prem datacenter including direct connected and VPN
    - Hub and spoke connection
    - Cross account and cross region (using Resource Access Manager)
    - Route tables = limit which VPC can talk with other VPC
    - Supports IP Multicast ( only AWS service )
    - ECMP
      * Equal Cost Multipath routing
      * Routing strategy to allow forward packet over multiple best path
      * Use case:
        + Create multiple site-to-site VPN connections to increase bandwidth of your connection to AWS
* Amazon VPC
  + Overview
    - Virtual Private Cloud
    - Can have multiple VPC in a region
      * Soft limit max 5 per region
    - IP Ranges
      * Max CIDER per VPC is 5
        + Min /28 = 16 IP addr
        + Max /16 = 65536 IP Addr
        + Only private IP addr allowed

10.x.x.x

172.16.x.x

192.168.x.x

* + - * + CIDR should not overlap other VPC or corp net
      * AWS reserves first 4 IP address and last IP for each subnet
        + Example

10.0.0.0/24

10.0.0.0 = Network address

10.0.0.1 = VPC Router

10.0.0.2 = Reserverd for mapping to Amazon provide DNS

10.0.0.3 = reserved for future use

10.0.0.255 = broadcast address (broadcast is not supported in VPC so the address is reserved and cannot be used)

* + - Default VPC
      * Each account has one
      * EC2 instances are started here by default
      * Has internet connectivity
      * All ec2 instances have IPv4 public in this VPC
      * Public and private DNS name
    - Internet Gateway
      * Allows resources in VPC to connect to internet
      * Scales horizontally and is highly available and redundant
      * Must be created separately from VPC (not during VPC creationg)
      * One VPC can only be attached to one IGW and vice versa
      * On their own IGW do NOT allow internet access
        + Route tables to allow traffic to and from the internet
    - Bastion Host
      * EC2 instance in public subnet of VPC that we connect to and allows us to pass thru to private subnet inctances
      * Should only have port 22 open from your IP address
      * Private EC2 instances should allow 22 from bastion host
    - NAT Instances
      * Outdated
      * Allow EC2 instances in private subnets to connect to the internet
      * Must be launched in the public subnet
      * Must disable Source destination check
        + Nat instance must be able to send and receive traffic whenthe source or destination is not itself (acts on behalf of private EC2 instance)
      * Must have elastic IP attached to it
      * Route tables must be configured to route traffic from private subnets to the NAT instance
      * Preconfigured Amazon Linux AMI
        + EOL Dec 31 2020
        + Not highly available / resilient

You have to configure all of this

* + - * + Internet traffic depends on EC2 instance
        + Manage security group rules

Allow SSH from your home network

Allow HTTP/HTTPS traffic from Private subnets

All HTTP/HTTPS out to internet

* + - NAT Gateway
      * Replacing NAT Instances
      * Aws managed
      * Higher bandwidth,
      * high availability
        + redundant within AZ
        + should have one in another AZ for high availability
      * Pay per hour for usage and bandwidth
      * NATGW created in a specific AZ with Elastic IP
      * Can’t be used by EC2 instances in same subnet
        + Requires an IGW ( Private Subnet =>NATGQ(public subnet) =>IGW
      * 5 Gbps bandwidth auto scale to 45 Gbps
      * No Security Groups to manage / required
    - VPC DNS Resolution
      * EnableDNSSupport
        + Defaults to true

Allows DNS resolution from Route 53 resolver server to VPC

Queries Amazon Provider DNS server at 169.254.169.253 or the reserved IP address at the base of the VPC IPv4 network range plus two

* + - * + False

Must have your own DNS server within the VPC

* + - * EnableDNSHostnames
        + Default

True = Default VPC

False = Newly created VPC

* + - * + Requires EnableDNSSupport to be true
        + True = assigns public hostname to EC2 instance if it has a public IPv4
        + When to use

If you use custom DNS domain names in a privete hosted zone in rout 53, you must set both of these

* + - Security Groups and NACLs
      * Security Group
        + Apply to resource
        + Stateful

Whatever is allowed in is allowed out

Whatever is allowed out, answer is allowed in

* + - * + Allow rules only
        + All rules evaluated
      * NACL
        + Apply to Subnet
        + Stateless

Outbound rules need to be allowed

Inbound rules need to be allowed

* + - * + Allow and deny rules
        + One NACL per subnet
        + Rules are ordered lower number looked at first

First rule match wins

* + - * + Last rule Asterisk deny all
        + Newly created NACL will deny everything
        + NACL are great at blocking specific IP
        + Default NACL

Accepts everything inbound/outbound with the subnets its associated with

* + - * + Ephemeral Ports

Clients connect to server on defined port

Expects a response on an ephemeral port ( port on client machine )

Diff OS use different ports

These ports need to be opened on NACL or return info will not happen

* + - VPC Endpoints
      * Private endpoint within VPC to connect to AWS resources without using the public internet
      * Also called AWS Private link
      * Redundant and scal horizontially
      * Remove need to IGW, NATGW, …
      * Troubleshooting
        + Check DNS Setting Resolution in VPC
        + Check Route tables

Must have route to resource

* + - * Types
        + Interface Endpoints

Provision ENI as an entry point to AWS Service

Must attach Security Group

Supports most AWS Services

* + - * + Gateway Endpoints

Gateway must be used as a target in a route table

Supports both S3 and Dynamo DB

* + - VPC PrivateLink – VPC Endpoint Services
      * Powers VPC Endpoint
      * Most secure and scalable way to expose a service to a lot of VPCs (own or customer accounts)
      * Requires
        + Network load balacer

On your service VPC

* + - * + ENI

On your client VPC

* + - * If NLB is in multi zones, then ENI needs to be in those multi Zones to create fault tolerance
    - Reachability Analyzer
      * Net diag tool to troublehoot network connectivity between two endpoints in your VPC
      * Builds a model of the network configuration then checks reachability based on these configs with no packets
    - VPC Peering
      * Privately connect two VPCs usein AWS network
      * Behave as they are in the same network
      * CIDR must be diff
      * Not transitive
      * Must update route tables in each VPC to ensure instances can commo with each other
      * Good to know
        + Peer between VPC in different Accounts and reagions
        + You can reference a security roup in a peered VPC
    - VPC Flow Logs
      * Capture info about IP Traffic going into your interfaces
        + VPC Flow Logs
        + Subnet Flow Logs
        + Elastic Network Interface (ENI) Flow logs
      * Helps to monitor and troubleshoot connectivity issues
      * Flow log data can go to S3 / cloudwatch logs
      * Capture info for AWS managed interfaces
      * Can be used for analytics on usage patterns or malicious behavior
      * Query VPC flow logs using athena on S3 or cloudwatch logs insights
    - Site to Site VPN
      * VPN between VPC and Corp datacenter
      * Requires
        + Virtual Private Gateway

VPN Concentrator on the SWS side

Created and attached to the VPC from which you want to create the site to site VPN

Possibility to customize the ASN (autonomous system number)

* + - * + Customer Gateway

Software or physical device in corp datacenter

Can be public IP or private IP (with NAT-T device for internet access)

* + - * + Must enable route propagation for the VPGateway in the route table associated with your subnets
        + To ping security groups must allow ICMP
      * AWS VPN CloudHub
        + If you have more than one Datacenter connect all of them via VPN to AWS VPNGatewa and with cloudhub all data centers can talk to each other over VPN
        + Dynamic route and route tables required
    - Direct Connect (DX) and Direct Connect Gateway
      * Dedicated private connection from DC to AWS
      * Uses AWS direct connect Locations
      * Allows you to access public and private AWS resources over same connections
      * Requires
        + VP Gateway on your VPC
      * Benefits
        + Lower cost
        + Increase bandwidth
        + More consistent experience
        + Hybrid environments
        + IPv4 and v6
      * Direct Connect Gateway
        + Allows you to create a connection to AWS and then using the gateway connect to multiple VPCs in different reagions
      * Connection types
        + Dedicated Connection

1 Gbps and 10 Gbps

Physical ethernet port dedicated to customer

Request made to AWS first then completed by AWS direct connect partners

* + - * + Hosted Connections

50 Mbps, 500 Mbps up to 10 Gbps

Request made via AWS Direct Connect Partners

Capacity can be added or removed on demand

* + - * Lead times are often longer than 1 month to establish a new connection
      * No encryption but private
      * AWS + VPN with IPsec-encrypted is you want encryption
      * Resiliency
        + High Resiliency for critical workloads

Multiple direct connect locations

* + - * + Max resiliency

Two direct connect location each with two connections

* + - AWS Classlink
      * Deprecated
      * Run instances in single network shared with others
      * Replaced by VPC
      * Classic link is a link between EC2-classic and VPC
    - Transit Gateway
      * Transitive peering between thousands of VPC and on prem datacenter including direct connected and VPN
      * Hub and spoke connection
      * Cross account and cross region (using Resource Access Manager)
      * Route tables = limit which VPC can talk with other VPC
      * Supports IP Multicast ( only AWS service )
      * ECMP
        + Equal Cost Multipath routing
        + Routing strategy to allow forward packet over multiple best path
        + Use case:

Create multiple site-to-site VPN connections to increase bandwidth of your connection to AWS

* + - VPC traffic Mirroring
      * Capture and inspect network traffic in your VPC
      * Route the traffic to security appliances that you manage
      * Capture traffic
        + From (source) – ENIs
        + To (targets) – and ENI or a Network Load Balancer
      * Capture all packets or filter by what you are interested in
      * Source and target same VPC or diff VPC if peered
    - VPC IPv6
      * Every ipv6 is public no private range
      * Cannot disable IPv4
      * When enable EC2 instances get both IPv4 and IPv6 IP
      * Troubleshooting
        + If can’t launch IPv6 EC2 instance in subnet probably because IPv4 subnets ran out
    - Egress only Internet Gateway
      * IPv6 only
      * Outbound only , no inbound initiated calls
      * Must update route table
* Amazon VPC Traffic Mirroring
  + VPC traffic Mirroring
    - Capture and inspect network traffic in your VPC
    - Route the traffic to security appliances that you manage
    - Capture traffic
      * From (source) – ENIs
      * To (targets) – and ENI or a Network Load Balancer
    - Capture all packets or filter by what you are interested in
    - Source and target same VPC or diff VPC if peered

### Security, Identity, and Compliance:

* AWS Certificate Manager (ACM)
* Amazon Detective
* AWS Directory Service
* Amazon GuardDuty
* AWS IAM Access Analyzer
* AWS Identity and Access Management (IAM)
* Amazon Inspector
* AWS Key Management Service (AWS KMS)
* AWS License Manager
* AWS Secrets Manager
* AWS Security Hub
* AWS Shield
* AWS WAF

### Storage:

* Amazon Elastic Block Store (Amazon EBS)
  + Overview
    - Elastic Block Store
    - Network drive
    - Persist data
    - Can only be mounted to one instance at a time
    - Bound to specific AZ
    - Like a shared drive
    - Commo via network
    - Can be detached and attached to diff EC2 instance
    - To move to diff AZ must create snapshot first
    - Delete on termination attribute
      * Deletes the volume when EC2 instance is terminated
      * Root vol enabled by default
  + Snapshots
    - Do not need to detach volume ( but recommended )
    - Snapshot can be copied across regions
    - Amazon Data lifecycle manager
      * Creation, retention and deletion of EBS snapshots and EBS Backed AMI
      * Schedule backups, cross-account snapshots copies delete outdated backups
      * Can use tags to select resources
    - Fast snapshot Restore (FSR)
      * Snapshots stored in S3
      * Helps create vol from snapshot that is fully initialized at creation
      * No IO latency
      * Billed per minute ( expensive )
      * Can be enabled via DLM
  + Instance Store
    - High performance
    - Better I/O performance
    - Ephemeral
      * Not persistent
      * Loose data when instance is stopped
    - Buffer, cache, scratch data / temp content
    - Backups and replication up to you
  + EBS Volume Types
    - Types
      * Gp2 / Gp3 (SSD)
        + General purpose
        + SSD
        + Balanced price and performance
        + 1 GB to 16 TB
        + GP3

3000 IOPS – increase to 16000

125 MiB/s – increase to 1000 MiB/s

Independently set IOPS and throughput

* + - * + GP2

Small vol burst to 3000 IOPS

Size of vol and IOPS are linked max 16000 IOPs

* + - * Io1 / Io2 ( SSD )
        + Highest Performance SSD
        + Mission critical
        + Low latency
        + High throughput
        + Provisioned IOPS
        + Critical business apps with sustained IOPs
        + Apps that need more than 16000 IOPS
        + 4GB – 16TB
        + Max IOPS 64000 (Nitro EC2 instances)

32000 for other instances

* + - * + Can increase IOPS independent of size
        + IO2 have more durability and mor IOPS per GB than IO1 at same price
        + Support EBS multi attach

Io1 and io2

Attach same EBS vol to multiple EC2 in same AZ

With full read/write

Clusted use cases

App must manage concurrent writes

* + - * St1 (HDD)
        + Low cost HDD
        + Designed for frequently access, throughput-intensive workloads
        + Cannot be used as boot volume
        + 125 MB – 16 TB
      * Sc1 (HDD )
        + Lowest Cost HDD
        + Les frequently accessed
        + Cannot be used as boot volume
        + Cold HDD
    - Characterized by
      * Size
      * Throughput
      * IOPS
  + EBS Volume Resizing
    - Can increase size and IOPs (IO drives only)
    - Must repartition drive after resizing
    - Volume may go into long optimization phase (Defrag) but is still usable
    - Decrease vol
      * Must create new vol of smaller size
      * Attach and migrate data
      * Remove old bigger drive
  + EBS Migration
    - Locked to specific AZ
    - To move to another AZ / region
      * Snapshot volume
      * Copy to diff region if need be
      * Create a volume from the snapshot in the AZ of your choice
  + EBS Encryption
    - By default when create EBS Vol
      * Data at rest is encrypted
      * All data in flight moving between the instance tnd the volume is encrypted
      * All snapshots are encrypted
      * All volumes created from the snapshot are encrypted
    - Encryption and decryption handled transparently
    - Encryption minimal affect on latency
    - Keys from KMS
    - Copying an unencrypted snapshot allows encryption
    - Enable encryption
      * Create snapshot
      * Encrypt using copy
      * Create new vol from snapshot
      * Attach new vol
* Amazon Elastic File System (Amazon EFS)
  + Overview
    - Managed NFS
    - Multi-AZ
    - Multi instance can access
    - Highly available
    - Scalable
    - Expensive
      * Pay per use
    - Like mounting NFS (network share)
    - Use cases
      * Content management
      * Web serving
      * Data sharing
      * Wordpress
    - Uses NFS 4.1
    - Use security groups for access
    - Only works with linux
    - Encrypt at rest use KMS
    - Scale automatically
      * 1000 of connections
      * 10 GB plus throughput
      * Can grow to peta byte
    - Performance Mode
      * Set at EFS vol Creation time
      * Types
        + General Purpose
        + Max I/O

less latency, throughput, highly parallel

* + - * Throughput Mode
        + Bursting

1 TB = 50 MB/S burst up to 100 MB/s

* + - * + Provisioned

Set your throughput regardless of storage size

* + - * + Storage Tiers (lifecycle management feature)

Standard

Frequently accessed files

Infrequent Access (EFS-IA)

Cost to retrieve but less to store

* + EFS Access Points
    - Feature to manage application access to NFS environments
    - Enforce POSIX user and group to use when access the file system
    - Allows you to restrict access to directory within file system
    - Specify different root directory
    - Can restrict access from NFS Clients using IAM policy
  + EFS Operations
    - In place operations ( when EFS is attached and in use)
      * Lifecycle policy ( enable IA or change settings )
      * Througput Mode
      * Provisioned Througput Numbers
      * EFS Access Points
    - Operations that require migration using datasync
      * Migration to encrypted EFS
      * Performance Mode
  + EFS Cloudwatch Metrics
    - PercentIOLimit
      * How close volume is to IO limit
      * If near or at 100% migrate to MaxIO
    - BurstCreditBalance
      * The number of burst credits the file system can use to achieve higher throughput levels
      * Remember if no credits then cannot burst
    - StorageBytes
      * Size in bytes
      * Dimensions
        + Standard
        + IA
        + Total (standard+IA)
* Amazon FSx
* Amazon S3
  + Buckets
    - S3 buckets must be named unique within entire AWS – globally unique
    - Object storage
    - Fast, durable, highly available, key based
    - Not a file system
    - Defined at the region level
    - Naming convention
      * No uppercase
      * No underscore
      * 3-63 chars long
      * Not an IP
      * Start with letter or number
  + Objects
    - Stored in buckets
    - Key
      * Full path to object
      * Composed of
        + Prefix

Path excluding bucket name and object name

* + - * + Object name

Object name

* + - Value
      * Content of the body
      * Max object size is 5TB
      * Can only upload 5 GB at time
        + Multi-part upload is when you upload multiple 5GB chunks to make up a large file
    - Metadata
      * List of key/value pairs
      * System or user metadata
    - Tags
      * Useful for lifecycle and security
    - Version ID
      * Need to be enabled at the bucket level
      * Upload / save object with same name it will create new version
      * Protects against unintended deletes
      * Easy roll back to previous version
      * Notes
        + Any file that is not versioned prior to enabling versioning will have aversion of null
        + Suspending versioning does not delete the previous versions
      * Deleting older version or deleting delete marker is a permanent delete
* Amazon S3 Glacier
  + Storage class of S3
* AWS Backup
* AWS Storage Gateway

AWS Beanstalk

* PaaS
* Dev centric view of deploying an application on AWS
* Free but pay for underlying resources
* Managed service
  + Instance config / OS config
  + Handles infrastructure
  + Deployment strategy
* Architecture models
  + Single instance deployment
    - Good for dev
  + LB + ASG
    - Production or pre production
  + ASG
    - Non web apps in production
    - No LB
* Three components
  + Application
    - Collection of Beanstalk components ( configs )
  + Application Version
    - Iteration of your application code
  + Environment Name
    - Collection of AWS resources running an application version
* Deploy application versions to environment
* Can promote application versions to next environment
* Rollback to previous app version
* Full control over lifecycle
* Support for many platfoorms

## Out-of-scope AWS services and features

The following is a non-exhaustive list of AWS services and features that are not covered on the exam.

These services and features do not represent every AWS offering that is excluded from the exam content.

Services or features that are entirely unrelated to the target job roles for the exam are excluded from this

list because they are assumed to be irrelevant.

Out-of-scope AWS services and features include the following:

* Amazon API Gateway
* Amazon AppStream 2.0
* AWS Batch
* Amazon Chime
* Amazon Cloud Directory
* Amazon CloudSearch
* AWS CodeBuild
* AWS CodeCommit
* AWS CodeDeploy
* AWS CodeStar
* Amazon Connect
* AWS Deep Learning AMIs (DLAMI)
* AWS Device Farm
* Amazon DynamoDB
* Amazon DynamoDB Accelerator (DAX)
* Amazon Elastic Container Registry (Amazon ECR)
* Amazon Elastic Container Service (Amazon ECS)
* Amazon Elastic Transcoder
* Amazon EMR
* Amazon GameLift
* AWS IoT Button
* AWS IoT Greengrass
* AWS IoT Platform
* Amazon Kinesis
* Amazon Lex
* Amazon Lightsail
* Amazon Lumberyard
* Amazon Machine Learning (Amazon ML)
* Version 2.1 SOA-C02 10 | PAGE
* AWS Managed Services
* AWS Mobile Hub
* AWS Mobile SDK
* Apache MXNet on AWS
* Amazon Pinpoint
* Amazon Polly
* Amazon Redshift
* Amazon Rekognition
* AWS Schema Conversion Tool
* Amazon Simple Email Service (Amazon SES)
* AWS Snowmobile
* Amazon WorkDocs
* Amazon WorkMail
* Amazon WorkSpaces
* AWS X-Ray

Color Key

Need everything = Training course, AWS DOCs, etc.

Need to read AWS Docs and do lab

Need Lab

Need AWS Docs

Complete