AWS Certified Security Specialty Master Cheat Sheet

Services

A complete list of the AWS security services, and selected additional AWS services of relevance to security (in particular, the security specialist certification). Taken from the AWS product list as of March 2019; if a category isn't listed it's because I thought none of the services in that category are particularly applicable.

Particularly important services from an exam perspective are in **bold**.

Security service links are to their FAQ pages, as a useful source of information on particular use cases and constraints that might be examined. Other service links are to their main product pages, but the FAQ pages often have good information including a security section too.

Security

- Artifact
  - Generic AWS compliance docs
- Certificate Manager
  - Issuance can take a few hours
  - Email or DNS validation (CloudFormation only supports email validation)
  - Validates DNS CA Authorization records first
  - Certs are region-locked, unless CloudFront is used (w/ Virginia)
  - Private keys are KMS protected - CloudTrail shows services using KMS to get the keys
  - Private CA
    - Allows export of the private key, whereas public standard only integrates with AWS services
- Cloud Directory

  - Generic directory service - not Active Directory. Could be used for user/device management.
  - Encrypted at rest and in transit

- CloudHSM
  - Advertised as only suitable when you have contractual/regulatory constraints.
  - Only option for SQL Server and Oracle transparent database encryption (but not AWS RDS Oracle! only instances running on EC2. RDS Oracle only works with CloudHSM Classic). Also works with Redshift.
  - PKCS#11, JCE, CNG
  - FIPS 140-2 Level 3 certified
  - KMS can use it as a key store - see KMS section
  - Each instance appears as network resource in VPC; client does load-balancing.
  - [[HSM] Server] <-TLS-in-TLS-> [client] <-p11 etc-> [app]
  - HSM users authenticate with username + password
  - CloudTrail for provisioning API calls; CloudWatch Logs for HSM logs
- **Cognito**
  - User Pools
    - Free up to 50k monthly active users
    - OAuth user tokens
  - Identity Pools
    - Mapping between federated user IDs and Cognito user IDs. Per pool.
    - Grants temporary AWS creds (either directly from federation, or in exchange for a user pool token)
    - IAM Roles assigned based on: mappings defined for a user pool group / rules / guest
  - API Gateway has direct support for Cognito tokens (no need for identity pool)
  - Sync store - key/value store per identity
  - Common scenarios
  - Various soft limits e.g. API calls/s, groups/pool, etc. No limit on number of users.
- **Directory Service**

    - Works with EC2 (manage them via group policies), RDS SQL server, WorkSpaces, AWS SSO, and a few more obscure ones
    - Can assign IAM roles to AD users for AWS access
    - Managed Microsoft AD
        - Can join to existing AD with trust relationships
        - Or replace an on-prem AD by using Direct Connect or VPN
        - EBS volumes are encrypted. Deployed on two AZs. Daily backups.
        - Some high-priv operations not available. No remote access or powershell access. You get an OU and delegated admin account for it.
    - AD Connector
        - Proxy for a specific list of AWS services through to on-prem AD.
        - Notably works with: SSO; management console; EC2 Windows (join domain)
    - Simple AD
        - Samba backend. Like Managed Microsoft AD but less features and smaller resource limits.

- Firewall Manager

    - Centrally manage WAF rules across CloudFront and ELB Application Load Balancers via Organizations
    - (not NACLs or Security Groups)

- **Guard Duty**

    - Uses CloudTrail, VPC Flow Logs, and DNS Logs (if EC2 instances are configured to use Route 53 resolvers - the default). Doesn't require you to enable them!
    - ^^ meta-data, + AWS' threat intelligence - domains & ips, + ML
    - Pricing per volume of data analyzed
    - Looks for reconnaissance, (ec2?) instance compromise, account compromise
    - Findings -> GuardDuty console (for 90 days) + CloudWatch Events. Findings in JSON format similar to Macie & Inspector
    - Regional. Can aggregate via CloudWatch Events to push to a central store
    - CloudWatch events -> SNS topic (-> email) / Lambda (->S3)

- **IAM**

- Users, Groups, Roles
  - Roles for EC2 instances
    - creds found in http://169.254.169.254/latest/meta-data/iam/security-credentials/<role / instance profile name>
    - To launch an instance, users need iam:PassRole for the relevant roles.
    - Can be attached at launch or later.
    - Auto rotation, built in support for obtaining the creds when using CLI & SDKs
  - Service linked role - predefined policy granting service what it needs; immutable trust policy.
  - Role trust policy: what principals (account/user/role/service/federated user) can sts:AssumeRole. IAM users/roles also need an identity policy that allow them to assume the role.
  - Assumed role ARN: `arn:aws:sts::AWS-account-ID:assumed-role/role-name/role-session-name`, where the session name might be the EC2 instance ID, or the IAM username, for example.
- Access keys
  - Rotate by creating second access key, start using it, check last used date of old one, make old one inactive, then delete it
  - Trusted advisor can look for overly long-lived access keys
- Policies
  - Resource based policies
    - Specifies a Principal.
    - Can't be managed policies - always inline.
    - Not actually IAM policies at all - just usually use the same policy language
    - Notable ones: Organizations (SCP); S3; API Gateway; Lambda; KMS
  - Identity based policies (aka IAM policies)
    - Attached to a user/group/role - implicit Principal
    - Limit of 10 managed policies can be attached
    - Versions - up to 5, you set which is the 'default' for customer managed policies. Inline policies don't have versions.
  - Permissions boundaries
    - Set the maximum permissions that an identity-based policy can grant to an IAM entity

        - Unlike SCPs, can specify resources and use conditions
- Service Control Policies (SCPs) - see Organizations
- Session policies - like a permission boundary, optionally passed programatically as part of AssumeRole*
- Evaluation logic - but there are special cases not listed here, e.g. KMS, S3
- Conditions
    - Operators
        - Date, Numeric, String, Bool, Binary (b64), IpAddress, Arn, Null (key:true - key doesn't exist, key:false - key does exist and isn't null)
        - operators are ANDed, multiple values in an operator are ORed
        - ...IfExists returns true if key doesn't exist
        - Set operators for keys with multiple values - ForAllValues:... ForAnyValue:...
    - All services: time, MFA, secure transport, user agent
    - aws:source{Vpc,Vpce (endpoint),Account,Arn,Ip}
    - aws:PrincipalOrgID - instead of listing lots of accounts, just use the Org. In resource policies - Principal:*, then this condition
    - aws:PrincipalTag/ - you can tag users and roles. Also service:ResourceTag and aws:RequestTag (control what tags users can use when tagging resources).
    - aws:PrincipalType
    - aws:RequestedRegion
    - aws:userid aws:username
- Policy variables
    - Use in resource element and string operators in conditions
    - Basically the same set of variables as global conditions. aws:username etc.
- (Not)Principal
    - AWS - users, roles, accounts
    - Federated - just "this principal authenticated with this provider" - no info on the role
    - Service - in trust policies
    - AWS:* - IAM identities (not services)

        - NotPrincipal rarely, and not with Allow as v fragile. NotPrincipal+Deny acts like a whitelist due to policy eval rules.
    - NotAction - matches everything except the list of actions. With Allow is very broad - combine with a resource constraint to make it more selective.
    - Resource
        - Wildcards - *? - don't span segments
        - NotResource + Deny: blacklist. NotResource + Allow: risky - allows all others incl. future ones.
- Access advisor
    - When did an entity last use a permission
    - For each of User, Group, Role, and Policy
- Federation
    - SAML
        - Users gets SAML assertion from their IdP portal, uses STS to exchange it for temporary creds.
        - IdP maps users/groups to roles.
        - Requires config info including keys registered with both the IdP and AWS IAM
        - Use AWS SSO to access the console.
    - Web identity federation - just use Cognito. IAM does support it natively too though.
    - Active Directory - use Directory Service, setup roles that trust DS, assign users or groups to roles
- Service support
    - Of interest are services that have resource-based policies, services that don't have resource-level permissions, and services that don't support temporary creds
    - Notable resource-based policies: ECR; Lambda; S3 & Glacier; KMS; Secrets Manager; API Gateway; VPC endpoints; SNS; SQS; SES
    - Notable ones missing resource level permissions: CloudFront (no resource policies either)
    - ~everything that matters supports temporary credentials
- Temporary credentials
    - Can't be revoked, but you can revoke an IAM user if they created the temporary creds, which invalidates them.
    - Include a token as well as access key & secret key. Token is appended to requests (header/query param)

        - Not regional
        - You can use AssumeRoleWithWebidentity as a less-featured alternative to Cognito w/ your users
    - Multifactor
        - No support for SMS any more.
        - U2F, virtual TOTP, hardware TOTP provided by AWS.
        - Root user can recover from lost second factor by verifying email address + phone number ownership.
        - APIs can require it by adding condition statements to identity or resource policies using `aws:MultiFactorAuthPresent` or `aws:MultiFactorAuthAge` (time since factor seen). Users then call STS to get temporary credentials that allow them to use the API. Doesn't work with root or U2f.
        - Doesn't work with federation

- Inspector
    - Rules packages
        - Predefined only.
        - Network: Network Reachability
        - Host: CVEs; CIS Benchmarks; Security Best Practices (OS config incl remote access); Runtime Behavior Analysis (protocols, ports, software config)
    - Template
        - Rules packages (predefined only), target EC2 instances, SNS topic
    - Network reachability + host config (CVEs in package manager installed software, CIS benchmarks for popular OSes)
    - Agent required for host config
    - Network reachability: enumerates what ports are accessible from outside of a VPC (+ what process listening on those ports, with agents)
    - Service linked role to enumerate EC2 instances and network config
    - Simple schedule in template, or more advanced via CloudWatch events / custom use of API

- **KMS**
    - Key policies
        - Required. Also different evaluation logic to standard IAM - if the key policy doesn't allow, then the request is denied regardless of identity policies.

        - Resource: "*" - this CMK
        - Principal: accounts/users/roles/services. Not groups! Have to use IAM identity policies to manage access via groups (or group -> assumerole).
        - Default policy for API-created CMKs allows `kms:*` for the account / root user. This ensure it doesn't become unmanageable, and also *enables* identity based IAM policies - without it IAM policies are ineffective.
        - Default policy for console created keys also allows you to specify:
            - Roles/Users who are Key Administrators, who can manage it - incl change its policy.
            - Roles/Users/other AWS accounts who are Key Users. They can encrypt/decrypt/generatedatakey, and manage grants for AWS services using the `kms:GrantIsForAWSResource` condition.
    - IAM/identity policies
        - Required for non-key specific tasks list ListKeys, ListAliases, and CreateKey
        - Required to use the console
    - Bunch of KMS-specific condition keys that can be used in either policy type.
        - `kms:ViaService` to prevent direct API use or block specific service use. All AWS managed CMKs use it to restrict access to the creating service.
    - Grants
        - Another resource-based policy attached to keys.
        - Allow-only, no Deny.
        - "grantee principal" - who can use the CMK.
        - "retiring principal" - who can revoke the grant
        - Actions: drawn from using the key, and creating further grants
        - Grant tokens: passed back when creating a grant, allows grantees to use the grant even before it has fully propagated. Not secret, no security impact, just practical.
    - Key usage -> CloudTrail
    - AWS services use wrapped data keys with KMS - 'envelope encryption'
    - APIs expose raw encrypt/decrypt operations, <4kb
    - CMKs
        - AES-256
        - CMKs are stored in HSMs (140-2 level 2)

        - AWS managed CMKs you have no control over. Customer managed ones you can set policies.
        - Imported CMKs can be deleted immediately and can have an expiry time.
        - 1000 CMKs per region
        - Keys are region-specific. For a multi-region solution, encrypt a single data key under CMKs in different regions.
        - Customer controlled CMKs can be enabled/disabled
        - Automatic annual key rotation can be enabled for customer controlled keys that don't use imported key material.
    - Custom key store
        - Uses CloudHSM
        - Can't import or automatically rotate keys - otherwise the same management as normal key stores
        - Only for customer managed CMKs
        - You're responsible for availability
        - Manual rotation: create key and remap key alias
    - CloudHSM
        - Single tenant 140-2 level 3 HSM - compliance
        - CloudHSMs appear in a VPC
        - Audit options beyond CloudTrail - CloudHSMs log locally and copy to CloudWatch
        - PKCS11 etc interfaces (as well as using as a custom key store)
    - Each region has a FIPS 140-2 validated endpoint (uses openssl fips module) and a standard endpoint.
    - AES-128 or AES-256 data keys
    - Crypto operations accept an optional *encryption context*, which is used as additional authenticated data (AAD) in the operation. If differs then decryption fails. Included in CloudTrail logs. Example used by S3:

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::bucket_name/file_name"
},
```

- Macie
    - Classifies data in S3.
    - Personally Identifiable Information (PII), Personal Health Information (PHI), regulatory documents (legal, financial), API keys and secret key material
    - Watches policy and ACL changes
    - Watches access patterns via CloudTrail

    - Alerts on CloudWatch Events, Lambda, and Macie dashboard
    - Primarily English

- **Organizations**
    - Organizational Units (OUs) divide up the 'administrative root'
    - Accounts can only be in one OU, and OUs can only be in one OU. But they can be nested up to 5 levels.
    - Service Control Policies (SCPs)
        - Which IAM policy Actions can be used in the account.
        - Applied to the root, to an OU, or to an account
        - Implicit and explicit Deny.
        - All statements: Version, Statement, Sid, Action, and Effect:Allow/Deny
        - Allow statements: no conditions, Resources must be '*'
        - Deny statements: support conditions and resources and NotAction
        - No principal - implicitly the accounts it's applied to
        - Is a whitelist, but can simulate a blacklist with Allow Action:'*' and another Deny statement
        - FullAWSAccess (allow *) is automatically attached to the root and new OUs. You can remove it.
        - Use policy simulator in member accounts to test effect
    - Trusted access
        - service-linked roles get created in member accounts as needed. Authorized via master account.
        - CloudTrail can create an organizational trail, for all events in all member accounts. Member accounts can't modify it.
    - Landing Zone account structures, incl logging & security accounts

- Secrets manager
    - Also see: Systems Manager Parameter Store - no rotation features, but free.
    - Automatic rotation for AWS RDS, DocumentDB, Redshift
    - Lambda functions to rotate other types
    - 4kb limit on secrets (JSON docs)
    - Encryption at rest via KMS. (for cross-account access to a secret, must use a custom CMK that the principal in the other account can use)
    - Policies

        - Resource-based (action+principal) and identity-based (action+resource) policies.
        - `arn:aws:secretsmanager:<region>:<account-id>:secret:optional-path/secret-name-6-random-characters`

```
  {
      "Sid" : "Get current TestEnv secrets",
      "Effect": "Allow",
      "Action": [ "secretsmanager:GetSecretValue" ],
      "Resource":
"arn:aws:secretsmanager:<region>:<account_id>:secret:TestEnv/*",
      "Condition" : {
          "ForAnyValue:StringLike" : {
              "secretsmanager:VersionStage" : "AWSCURRENT"
          }
      }
  }```
```

        - Condition keys include `secretsmanager:ResourceTag/<tagname>`, `secretsmanager:VersionStage`
        - Configuring rotation requires creating and assigning a role to a Lambda function, which needs e.g. IAMFullAccess

- Security hub
    - Regional - findings don't cross regions
    - Multi-account support
    - Findings from Guard Duty, Inspector, Macie, third party, and self-generated against CIS standards
    - Insights: collections / filters of findings
- Shield
    - Standard - integrated into existing services. Not a stand-alone service. Netflow monitoring & TCP/UDP protection.
    - Advanced
        - Layer 7 protection, WAF rule creation
        - CloudFront integration - can protect non-AWS origins
        - CloudWatch metrics notifications of attacks
        - Global threat environment dashboard, see overall stats for the whole of AWS
        - AWS DDoS team support
- SSO
    - Free
    - Primary use case: manage multi-account access with Organizations.

- Additional use case: SSO to other applications via SAML 2 (custom or a bunch of built-in integrations)
- IAM identity provider created in member accounts for SSO. Also service-linked roles created to allow SSO to manage Roles
- Sign-ins logged to CloudTrail
- Directories
  - Native directory - default. Create users & groups within SSO
  - AWS Directory Service - Managed AD & AD Connector (not simple AD)
  - Only a single directory can be connected
- Permissions sets
  - collections of policies.
  - Implemented as Roles in member accounts.
  - Limit of 20 per account.
  - Ref 10 AWS managed policies, or use an inline policy
- Control access by mapping users/groups (from the attached directory) to permissions sets & accounts. This data is held in SSO, not the directory.
- No API!
- For CLI access, SSO user portal gives you temporary creds for the Roles you have access to

- WAF
  - Conditions
    - Inspect: IP addresses (+ region mapping), HTTP headers, HTTP body, URI strings
    - Match against: SQL injection, cross-site scripting, regex, strings, IP ranges, regions, sizes.
  - Rules
    - Comprise a number of conditions ANDed together
    - Rate based rule - 5 minute period for given IP, e.g. to protect against DDoS or login brute forcing
    - Need conditions for normal rules, but they're optional for rate-based rules (no condition=all requests count)
    - Managed rules from Marketplace sellers.
  - Web ACLs
    - Collection of rules, ORed together
    - Actions per rule: allow, block, or count (for testing)
    - Default action if no rule matches

- Associate Web ACLs with CloudFront, ALB, and API Gateway instances which will then proxy requests via WAF and act on result
- Also see Firewall Manager and Shield (Advanced)

Analytics

(mostly of interest for their application to logs)

- Athena
    - SQL queries over data in S3 after you define a schema. Including (optionally compressed) JSON & CSV
    - Integrates with Glue's Data Catalog - a more featureful version of Athena's built in Data Catalog which supports fine-grained permissions.
    - Charged per query (volume of data scanned)
    - Security model uses both athena:* permissions for queries and data models, and then the underlying S3 permissions
    - Can query encrypted data that uses S3 or KMS managed keys. Can encrypt results.
    - Athena is better than Redshift for querying smaller datasets without pre-processing.
    - CloudTrail can automatically create Athena tables for you, and AWS are keen to push Athena as an ideal CloudTrail analysis tool. Other good candidates: VPC flow logs (if sent to S3), CloudFront, ELB.
- Elasticsearch service
    - IAM auth for management, ES APIs, and resource-based policies down to index level
    - Resource based policies can allow specific IP addresses
    - Kibana auth via Cognito
    - Can configure public or VPC endpoints
    - Ingress via Kinesis Firehose, Logstash, or ES's index/bulk APIs
    - KMS integration for data at rest
- Glue
    - "Select a data source and data target. AWS Glue will generate ETL code in Scala or Python to Extract data from the source, Transform the data to match the target schema, and Load it into the target. "
    - Sources: S3, Redshift, and RDS and other databases
    - Loading into other services for querying (e.g. Athena, Redshift)

- Kinesis
    - Ingest and analyse various data sources, notably logs
    - Data Firehose
        - "capture, transform, and load streaming data into Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, and Splunk"
        - Create delivery stream, with optional Lambda function to transform the data
        - Configure producers to send data to Kinesis with the Kinesis Agent (which monitors log files) or Firehose API
        - Source integrations: CloudWatch Logs subscription filter; CloudWatch Events rule with Firehose target; Kinesis Data Streams.
        - Configure an IAM role that it assumes to access e.g. S3 or Elasticsearch
        - Manage delivery frequency with buffer size or interval
- Redshift (see Database section)

Application Integration

- SNS
    - Pub/sub.
    - Sources include: SNS API, Lambda, ELB, S3, databases, Code*, CloudWatch, Inspector, and others
    - Destinations: Lambda, SQS, webhooks, SMS, email
    - Subscribers have to validate - a challenge message is first sent
- SQS
    - Polling, vs SNS's push mechanism
    - Standard queues might reorder messages or deliver them multiple times
    - Has its own resource-based security policy, that predates IAM? Looks similar to IAM policies. Only resource is a queue.
    - Can subscribe to SNS topics
    - Can trigger Lambda functions on message receipt
    - Uses KMS for optional encryption

Compute

- **EC2**
  - AMIs
    - LaunchPermission attribute - which *accounts* can use the AMI.
  - Keypairs
    - Create or import - 2k RSA.
    - Independent of instances, but each instance is associated with 1+ keys
    - Linux: it's just an SSH key
    - Windows: upload the private key to the ec2 console to decrypt the default admin password so you can RDP in...
    - Subsequent management: tinker with the `authorized_keys` file
  - Resources and condition keys
  - Instance store - hard disk attached to the instance; reset when the instance is stopped. Not encrypted - could use host software disk encryption for a temporary data partition.
  - Instance profile - credentials for a role available to the instance (see IAM section)
- Elastic Container Registry (ECR)
  - IAM access control for pulling & pushing images - identity & resource based
  - Repository policies - e.g. to allow other accounts to pull
  - Images encrypted at rest by default with S3 SSE; HTTPS access
- Elastic Container Service (ECS)
  - Tasks: set of containers that are placed together.
  - Containers run on customer-controlled EC2 instances in a VPC, or are Fargate managed.
  - Networking options:
    - none
    - bridge - docker's virtual network
    - host - tasks get the host's network interface
    - awsvpc: Task network interfaces are normal ENIs so all the VPC properties apply: exist in a subnet, have security groups, have flow logs. Also means each container can have its own security group & IP, vs host networking where all the containers on one host share interfaces.
  - Tasks are configured with an execution role they use to access services
  - Can send logs to CloudWatch

    - Fargate launch type
        - Must use awsvpc network mode, CloudWatch logs
        - Uses Firecracker under the hood (definitely not in scope of the exam, but an interesting topic!)

- Lightsail
    - Like an entirely separate cloud offering within AWS, with extremely limited features. DigitalOcean competitor.
    - No VPC - separate management of exposed ports
    - Hopefully not in the exam :)
- Elastic Beanstalk
    - Management wrapper around EC2, S3, EBS, RDS
    - Publicly available by default - configure to use a VPC to limit access
    - Beanstalk service role to manage other services. Instance profile - role used by instances to get the app, write logs, etc
    - Logs stored locally, can be configured to use CloudWatch Logs
- Fargate - see ECS
- **Lambda**
    - Logs to CloudWatch
    - Execution role
        - assumed to run
        - at minimum CloudWatch logs creategroup/createstream/putevents
        - Potentially also XRay write, SQS/Kinesis/dynamodb read to get the event data
    - Resource policies
        - Resources: functions, their versions and aliases, and layer versions
            - `arn:aws:lambda:region:123456789012:function:my-function`
            - `arn:aws:lambda:region:123456789012:function:my-function:1` - version
            - `arn:aws:lambda:region:123456789012:function:my-function:TEST` - alias
        - Use to give other services (principal: service: sns.ama...) and other accounts (principal: aws: account-arn) permission to use them

        - The console updates function policies automatically when you add a trigger to give the triggering service access
    - Identity policies
        - nice examples: ARN pattern so users have to include their username in function names; have to include a logging layer
        - To give users the ability to create functions with limited permissions, constrain what roles they can iam:PassRole on.
        - To give users the ability to add resource permissions to functions so they can be invoked, but only from specific sources, check lambda:Principal in a condition
    - VPC access
        - Can access resources in a VPC if subnet + security group is specified.
        - No internet access unless there is a NAT in the VPC.
        - No AWS service access unless there is internet access or VPC gateways
        - Role needs ability to create network interfaces in each subnet (and VPC must have ENI capacity & subnets must have spare IPs)

- Elastic Load Balancing (ELB)

    - Integrated with Certificate Manager to terminate TLS. Can also upload certs to IAM and configure ELB to use them from there.
    - Can specify which of several predefined cipher-suites - 'security policies' - to support
    - Application Load Balancer (ALB) - HTTP/HTTPS
        - In a security group
        - Integrated with WAF
        - Authentication: integrates with Cognito and supports Open ID Connect. Redirects users to IdP authorization endpoint, then adds headers with signed JWT containing user info.
        - Can have a Lambda function as a target. Transforms JSON response to HTTP. Function policy needs to allow `elasticloadbalancing.amazonaws.com` to InvokeFunction
        - Can enable access logging to an S3 bucket
    - Network Load Balancer - TCP/TLS
        - Doesn't support Server Name Indication (SNI)
        - 2k RSA certs only (ALB is more flexible)
        - Creates a (read only) network interface in a subnet in each AZ you choose. Not in a security group - instance security groups

must allow traffic from its IP address and from client IP addresses
  - (Classic)
  - Logs to S3

Customer Engagement

- Simple Email Service (SES)
  - potentially incident notification, but SNS probably more appropriate
  - Can receive mail, which can be encrypted using a KMS protected key. SDK available to support decryption.
  - TLS API or TLS SMTP connection (port 587), also supports STARTLS and DKIM, and can work with SPF and DMARC

Database

A comparison and summary of some of the security aspects of the various database offerings:

Database	Transport encryption	Encryption at rest	Audit	DB Authentication	DB Authorization
RDS	Rooted at global RDS certs, configuration is per-engine docs	KMS; TDE w/ SQL Server and Oracle - RDS managed key (used to be CloudHSM Classic)	per-engine log files	per engine user accounts - SQL	per engine - SQL
DynamoDB	Standard AWS	KMS	CloudTrail, excl.	IAM only. Cognito	IAM identity policies -

Database	Transport encryption	Encryption at rest	Audit	DB Authentication	DB Authorization
	HTTPS endpoint		Get/Put docs	possible. docs	resources & condition keys docs
Redshift	ACM managed certificate, redshift specific root docs	KMS; CloudHSM Classic	S3 docs	DB user accounts - SQL; IAM with custom drivers docs	SQL
Neptune	Publicly trusted Amazon root; mandated for some regions docs	KMS	Console docs	User accounts; or a limited IAM identity policy mechanism + request signing docs	Engine-specific; or broad access if using IAM
Aurora	Rooted at global RDS certs, configuration as per mysql/postgres docs	KMS	mysql -> CloudWatch Logs docs	User accounts; or an IAM authenticated API to obtain short lived passwords to connect docs	mysql/postgres - SQL

Database	Transport encryption	Encryption at rest	Audit	DB Authentication	DB Authorization
DocumentDB	Rooted at global RDS certs, configuration as per MongoDB docs	KMS	CloudWatch Logs docs	MongoDB user accounts	MongoDB standard

- DynamoDB
    - Optional encryption at rest integrated with KMS
    - Main resource is a table. No resource based policies. Full access to a table requires access to not just the `table/<name>` resource, but also `table/<name>/*`
    - Some predefined policies: `AmazonDynamoDBReadOnlyAccess`, `AmazonDynamoDBFullAccess` - custom policies with resource constraints are better
    - Several condition keys for fine-grained access including: `dynamodb:LeadingKeys`, `dynamodb:Select`, `dynamodb:Attributes`
    - Example fine-grained permission: you can only access items where the partition key matches your own (web identity) user ID, by using LeadingKeys and a substitution variable.
    - Get and Put API calls are not logged to CloudTrail - management things are like describe, list, update, create
    - Has a VPC endpoint you can use
    - Integration with Cognito: identity pool with roles configured; roles have appropriate policy to (a) allow cognito to assume them and (b) perform desired DynamoDB actions.
- RDS
    - IAM controls database instances. Each instance type has its own permission model for managing the database - a master user is created with the instance.
    - Lots of different resources. The main one is an instance - `db` in the arn. No resource based policies.
    - 'RDS Encryption' - encryption at rest, set during creation, uses KMS. Covers database, backups, replicas, snapshots.

    - Transparent data encryption for SQL Server and Oracle with CloudHSM
    - There's a single root for all RDS database TLS certs; each engine uses its own method for connecting over TLS
    - Manifests as network interfaces in subnets with security groups attached to the interfaces. You specifc a "db subnet group" - a collection of subnets which it can use to put interfaces in.
    - "Publicly accessible" option controls whether there is a publicly resolvable DNS name for the instance. Still needs appropriate security group rules.

- Redshift

    - Cluster management with IAM.
    - Database user accounts for DB permissions (SQL).
    - With custom Amazon Redshift JDBC or ODBC drivers, you can authenticate via IAM and get temporary DB user creds. Gives access to existing users or creates new users (groups specified via claims).
    - Lots of resources, main one is a cluster. No resource based policies. Managed policies to give access to all resources - `AmazonRedshiftFullAccess` and `AmazonRedshiftReadOnlyAccess`
    - Cluster are associated with 1+ security groups. Doesn't appear as an interface in a subnet. Contrast with RDS and DynamoDB - all different combos of network access control.
    - Audit logs, disabled by default, -> S3 (as well as the standard CloudTrail logs). Bucket policy has to allow putobject and getacl to a specific user from a redshift AWS account that varies by region: `arn:aws:iam::<redshift regional account id>:user/logs`. If creating the bucket via the console, it does that for you.
    - Optional encryption at rest. With KMS or CloudHSM Classic (only). Big symmetric encryption key heirarchy.

- Neptune

    - HTTPS access
    - Encryption at rest with KMS
    - Interface appears in at least two subnets spanning two AZs in a VPC, interfaces have security groups.
    - CloudTrail events appear as though they are from the RDS service not Neptune - it shares some underlying management infrastructure.
    - Optional audit logs, view or download from the console (no other service integrations, strangely)

- IAM for management. Permissions are a subset of rds permissions all the actions are `rds` actions. Can constrain to just neptune with a condition of `rds:DatabaseEngine = graphdb`
- Has a very unique hybrid model where you can authenticate with IAM, and define identity policies that allow access. Limited - no condition keys, no fine grained access (only a single `neptune-db:*` action). Pretty confusing when compared to the previous point. HTTP requests then need to be signed with standard AWS v4 signatures that you construct yourself.

- Aurora
  - The same as the other RDS engines, except:
  - Supports IAM database authentication, similar to Neptune. Attach identity policy to IAM principals that allow `rds-db:connect` for a resource that is a particular database user you create in particular way in the DB. You manage user permissions within the DB as per normal - IAM is just for authentication. You get a 'token' from the RDS API by specifying the db and user, then use the token in place of the user's password when connecting normally.
  - Uses normal VPC security groups to control access within a VPC. Has its own 'DB security group' to control access from outside the VPC - either security groups in other VPCs/accounts or the internet? The other RDS engines only use DB security groups in EC2 classic when a VPC isn't available.

- DocumentDB
  - Similar to RDS: TLS from the RDS root; KMS encryption at rest; master user + mongodb user mgmt; IAM identity policies for management; VPC security groups; endpoints on multiple subnets/AZs; cloudtrail
  - arns follow the RDS format
  - Auditing can be enabled to send events to CloudWatch Logs. Categories: connection, data definition language (DDL), user management, and authorization

Developer tools

- Code Pipeline
  - Resource-level permissions for pipelines, and their stages and actions.
  - Can integrate with GitHub via OAuth
  - CloudWatch Events for pipeline state changes - started, failed, etc.

- Supports interface VPC endpoint
- Trigger from, e.g.: CloudWatch Events (many options, e.g. S3 bucket upload, schedule), webhooks (e.g. github), manual
- Deploy to, e.g.: CloudFormation, S3, ECS, Service Catalog

End User Computing

- WorkSpaces
  - Supports EBS volume encryption for both root and user volumes
  - CloudWatch Event on user login
  - Uses AWS Directory Service for user authentication, works with any of Managed AD, AD Connector, and Simple
  - Can require Mac and Windows clients to use a certificate to authenticate a device to connect
  - WorkSpace network interfaces are associated with a standard VPC security group
  - Has some form of MFA support

Internet of Things

These sound like they should be in scope, but I suspect they're not as they're very niche.

- IoT Device Defender
- IoT Device Management

Management and Governance

- CloudFormation
  - Stacks
    - You can assign a service role, if you can iam:PassRole it. Anyone who can operate on that stack can leverage that role's permissions (even if they can't run it - they could modify it then someone else runs it!).
    - Otherwise the user/role that is using the stack needs to have permission to perform all the operations
  - StackSets
    - Custom administration role, with identity policies that constrain iam:PassRole for that role to control who can use it

        - Custom execution role, with limits on what resources it has action to, and a trust policy for specific administration role(s) in the admin account
    - Some interesting condition keys:
        - `cloudformation:ChangeSetName` e.g. enforce prefixes
        - `cloudformation:ResourceTypes` to control which resources can be involved in a stack
        - `cloudformation:TemplateUrl` e.g. can only create stacks from this URL (as oppoed to operating on an existing stack resource)

- CloudWatch

    - **Logs**
        - CloudWatch Agent can be installed on a host (e.g. via SSM) to push logs to CloudWatch Logs. Troubleshooting info.
        - Log group: a collection of log streams that share the same retention, monitoring, and access control settings
        - Log stream: a sequence of log events that share the same source
        - Logs last forever unless you set a retention period on a group
        - Subscription filters: define a filter pattern that matches events in a particular log group, send them to Kinesis Data Firehose stream, Kinesis stream, or a Lambda function.
        - Can export log groups (in a particular time range) to S3. Not real time.
        - Can receive events from other account by creating a 'destination' in CloudWatch, which references a receiving Kinesis stream? The destination has a resource-based policy that controls which accounts can write to the destination. CloudWatch Logs on the sender side can then stream to the other account.
    - Logs Insights
        - Limited query language for analysis and visualization of data in CloudWatch Logs
        - Much more powerful than the native CloudWatch Logs interface
    - Events
        - Rules that trigger from either event patterns or a schedule
        - Rules send JSON to one or more targets
    - Has other capabilities (metrics, alarms, scaling)

- **CloudTrail**

    - Also logs Cognito events, step function logs, and CodeDeploy

- Logs to S3 and/or CloudWatch Logs
- Without creating a trail, the event history shows 90 days but excludes various events including all read events
- A small number of services don't log to CloudTrail, notably SimpleDB
- Trails by default don't include data events (incl S3 object activity and Lambda execution). Can specify those resources you want to record.
- Trails are regional, but you can create a global trail which creates identitical trails in all regions. Limit of 5 trails per region.
- eventSource: what service produced the event.
- Can enable SNS notifications for when a new log *file* is produced
- Can set up CloudWatch metric filters for certain events to trigger a CloudWatch Alarm

- **Config**
  - Resource inventory, configuration history, and configuration change notifications
  - Configuration changes or deviations -> SNS, CloudWatch Events, console dashboard, S3
  - Regional, but can aggregate data across (a limited set of supported) regions and accounts. Can't centrally manage rules.
  - Inspects software running on SSM managed EC2 instances, incl OS version, installed apps, network config.
  - Configuration changes sent to 'delivery channel' - S3 bucket & SNS topic
  - Console provides a timeline view of configuration changes
  - AWSConfigRole is the managed audit role; also needs permisisons for the SNS topic & S3 bucket.
  - Rules
    - Continuously evaluate configs against rules
    - Retrospective and non-enforcing
    - Custom rules in Lambda
    - Soft limit of 50 active rules
    - Periodic (hourly to daily) or change-triggered. Change-triggered must be constrained by tag/resource type/resource id

- Control Tower
  - In preview at the time of writing - likely to become an important security service as it enables easier robust multi-account setups.

- Management Console
  - The web console!
- Service Catalog
  - Portfolio: collection of catalogs. Catalogs: collection of products. Product: CloudFormation template (with the usual optional CloudFormation parameters).
  - Portfolios can be shared across accounts.
  - Admin access control is via IAM. User access control is initially via IAM - You need ServiceCatalogEndUserAccess to use Service Catalog. It doesn't support resource-level permissions nor resource-based policies, which is weird. Portfolio access is instead managed within Service Catalog by associating IAM users/groups/roles with a Portfolio.
  - Launch role: a role that is used to run the templates, instead of the user having the necessary permissions. Don't think the user needs iam:PassRole to use it - so a way of constraining user of the permissions in the role.
- **Systems Manager (SSM)**
  - Group resources of different types together based on a query, e.g. an application.
  - Many features require the Agent installed - many AWS AMIs include it by default. EC2 instances need an instance profile for a role that has the necessary permissions to allow the agent to interact with SSM.
  - Insights dashboard - per resource group
    - Shows CloudTrail, Config, software inventory, and patch compliance
    - Can integrate CloudWatch dashboards, Trusted Advisor notificaitons, Personal Health Dashboard
    - Potentially useful for understanding baseline usage patterns to contrast with during an incident
  - Inventory - applications, files, network configurations, Windows services, registries, more
  - Automation
    - documents of tasks to run; scheduled, triggered, or manually launched
    - Approval feature - configure approvals required (via the console) before it continues
    - Documents can have roles, and users can have permission to run documents - nice restriction of privileges to particular tasks

- Run command
  - Sometimes called EC2 run command
  - Logs via CloudTrail
  - Can be triggered by CloudWatch Events
- Session Manager - browser based shell w/ IAM & CloudTrail
  - Can log session data to S3 and/or CloudWatch Logs
- Patch Manager
- State Manager - specify OS configuration, rollout schedule, compliance reporting
- Parameter store
  - Can be tagged + organized in a hierarchy.
  - KMS for encryption - users need KMS permissions to use the corresponding CMK (can restrict using a condition on kms:EncryptionContext to just particular parameters)
  - IAM resource per-parameter
  - 10k params per account
- Patch Manager and State Manager can operate on on-prem instances too
- Lots of resources, no resource-based policies
- The CloudWatch Agent can send SSM actions on the host to CloudWatch Logs

- **Trusted Advisor**
  - 7 free checks, all checks with appropriate support plan.
  - API; Console; Weekly notification email with summary of findings
  - Can exclude resources from all checks. Can't suppress individual checks.
  - Cost optimization, security, service limits, fault tolerance, performance
  - Security checks:
    - Security group open access to specific high-risk ports
    - Security group unrestricted access
    - Open write and List access to S3 buckets
    - MFA on root account
    - Overly permissive RDS security group
    - Use of cloudtrail
    - Route 53 MX records have SPF records
    - ELB with poor or missing HTTPS config
    - ELB security groups missing or overly permissive
    - CloudFront cert checks - expired, weak, misconfigured

        - IAM access keys not rotated in last 90 days
        - Exposed access keys on GitHub etc
        - Public EBS or RDS snapshots
        - Missing or weak IAM password policy

- Snow Family (see storage)

Mobile

- API Gateway (see network & content delivery)

Networking & Content Delivery

- API Gateway
    - Logs to CloudWatch
    - sigV4 signed requests with IAM; or Cognito User Pool token verification; or Lambda authorizers for other token verification
    - Can configure with a 'client-side' certificate that API gateway uses for authenticating its requests to backend servers
    - Resource based policies attached to API, the only action is `execute-api:Invoke`. Can use to allow cross-account access, or in combo with conditions to constrain access to specific VPCs / VPC endpoints / IP ranges etc. Rather complex logic for evaluating them in combo with identity policies.
    - Supports rate limiting requests from an IP
    - Private APIs - only accessible through VPC endpoints.
    - Private integrations - connect to non-public VPC resources behind the API. Create an ELB network load balancer in the VPC, API Gateway associates it with a 'vpclink' VPC endpoint
    - CORS - necessary to allow cross-origin requests; will need to be configured if using the default API gateway URLs rather than proxying via CloudFront, otherwise browsers won't honor requests to the API.
    - Integrates with WAF
- CloudFront
    - Optional access logs to S3 - bucket ACL configured to give the awslogsdelivery account full control. Metrics via CloudWatch.

  - Field level encryption - CloudFront can encrypt specific POST fields with a public key you've configured. Reduces exposure of sensitive data as it passes through the backend.
  - HTTPS: can configure HTTP, redirect to HTTPS, or HTTPS only for client side. For origin side can do HTTP, match viewer, or HTTPS.
  - To serve content from S3 *only* via CloudFront, create an 'origin access identity' for the distribution, then create a bucket policy that blocks public access and allows the special `"Principal":{"CanonicalUser":"<CloudFront Origin Identity Canonical User ID>"}`
  - Can only allow specific geographic regions based on IP
  - Can require signed URLs or signed Cookies - CloudFront creates keypairs for each "trusted signer" AWS account, and the account generates time-limited signed URLs or Cookies for clients to use.

- Route 53
  - Private DNS - create a hosted zone associated with at least one VPC.
- VPC PrivateLink - see VPC Interface Endpoints
- App Mesh
  - Envoy for ECS/EKS. Security is important if your app uses this, but unlikely to be in scope of the cert.
- Direct Connect
  - Dedicate WAN link to AWS
  - Alternative backend to Virtual Private Gateway instead of "vanilla internet"
  - Doesn't use encryption?
  - Virtual interfaces are either private - access to a VPC, or public - access to AWS public endpoints. Can have multiple interfaces per connection if its fast enough.
- Transit Gateway
  - "A hub that controls how traffic is routed among all the connected networks which act like spokes"
  - Instead of lots of (1:1) VPC peering relationships and lots of (1:1) VPN connections, connect each VPC to the single transit gateway and manage centrally
- **VPC**

- Spans all AZs in a single region
- Soft limit of 5 VPCs per region
- Has a CIDR, can have 4 additional CIDRs
- See example scenarios
- Policy resources and condition keys
  - Most resources support the `ec2:Vpc` and `ec2:Region` condition keys. Other notable ones listed below.
  - `arn:aws:ec2:<region>:<account>:internet-gateway/igw-id`
  - `arn:aws:ec2:<region>:<account>:network-acl/nacl-id`
  - `arn:aws:ec2:<region>:<account>:network-interface/eni-id` and `ec2:{Subnet,AvailabilityZone}`
  - `arn:aws:ec2:<region>:<account>:route-table/route-table-id`
  - `arn:aws:ec2:<region>:<account>:security-group/security-group-id`
  - `arn:aws:ec2:<region>:<account>:vpc/vpc-id` and `ec2:Tenancy`
- Network interfaces
  - Has one or more IP addresses, a MAC address, one or more security groups,
  - Can be moved between EC2 instances
  - Can't move the primary interface of an instance
- Egress options:
  - Internet Gateway
    - Attached to VPC
    - Interface must have a public address, but the gateway does NAT so incoming traffic is addressed to the interface's private address
  - Virtual Private Gateway
    - IPSec VPN attached to a VPC
    - Need a corresponding customer gateway in the other network(s)
    - Route table(s) need updating to point at customer gateway. Route propagation can do this automatically.
    - Security groups need rules to allow access from remote network
  - VPC Peering Connection
    - VPC peering can cross both accounts and regions, but is not transitive between VPCs
  - VPC Endpoints
    - To keep service traffic within AWS. No public IP needed.
    - Endpoint policies - resource policies that constrain what service actions are possible via that endpoint.

- S3 bucket policies can limit access to a specific endpoint or VPC using aws:sourceVpce and aws:sourceVpc, e.g.:

```
▪ {   "Sid": "specific-vpc-endpoint",
▪     "Condition": {
▪         "StringNotEquals": {
▪             "aws:sourceVpce": "vpce-1a2b3c4d"
▪         }
      },
```

- Similarly can use `aws:sourceVpce` in an identity policy for DynamoDB
- Gateway Endpoint
    - Gateway in the VPC that you route to with a special-case entry in route tables
    - S3 and DynamoDB only - they don't have interface endpoints
- Interface Endpoint (PrivateLink)
    - Elastic network interface with a private IP address
    - In a subnet and security group(s) - security group needs to allow outbound access to the service
    - Several services including EC2, ELB, SNS, CloudWatch, Systems Manager, and various Marketplace products.
    - Has an endpoint specific DNS hostname.
    - Private DNS allows you to use the normal hostname for the services, by creating a DNS zone in the VPC using Route53 that has a record for the service that resolves to the interface's private IP address.

- NAT Gateway
    - To prevent unsolicited inbound connections but allow outbound connections for instances without a public IP
    - Within a public subnet, in a specific AZ
    - The subnet's NACL applies, but NAT Gateways aren't in any security groups
    - Has an Elastic IP address
    - Connects to an Internet Gateway
    - Can be used by instances in a different (private) subnet in the same VPC
- Also see Transit Gateway

- Subnets
    - Within a single AZ
    - Can be shared across accounts!

        - CIDR is within the VPC's CIDR and can't overlap other subnets in the VPC. Must have IPv4 CIDR.
        - Associated with a route table for outbound traffic. Default to VPC's main route table.
        - Public subnet = route table includes an internet gateway. Otherwise called a private subnet.
        - Instances have a private IP and optionally (configured at subnet + instance level) either a public IP (random from AWS' pool) or an Elastic IP (persistent, owned by your account)
        - Instances with a public/elastic IP also get a public DNS hostname
        - Network ACLs
            - Each subnet has a NACL
            - What traffic can enter/exit a subnet
            - Stateless - must have explicit inbound and outbound rules - replies aren't special. For web-facing servers, need to allow outbound ephemeral ports e.g. 1024+ for all addresses
            - VPC default NACL is used for new subnets, its initial rules allow all traffic
            - Rules: Allow/Deny, dest port, src/dst addr, protocol.
            - Rules evaluated in order until one matches. Default deny (there's an immutable final deny rule that matches all).
            - Custom NACLs start with no rules (except the deny-all).
    - Route tables
        - Exist in the VPC. Subnets are associated with a single route table
        - The most specific route that matches is used
        - Always have unmodifiable local routes for in-VPC traffic
        - Need to have entries for gateways and VPC peering
        - New VPCs have a main route table. You can make a custom route table the main one.
    - Flow logs
        - to S3 or CloudWatch Logs
        - Log streams/files are per interface, but can be configured at VPC, subnet, or network interface level
        - Capture window: ~10 minutes after which a log entry is published
        - `<version> <account-id> <interface-id> <srcaddr> <dstaddr> <srcport> <dstport> <protocol> <packets> <bytes> <start> <end> <action> <log-status>`

        - Doesn't record: Amazon DNS requests (does record requests to a custom DNS server); 169.254.169.254 metadata; DHCP; traffic to the default VPC router
        - Identity policies only - no resource based policies
        - Flow logs service needs a role to assume so it can publish logs to S3 or CloudWatch, and users need iam:PassRole for the role
        - S3 Bucket policy must allow the service to PutObject + a bit more. Automatically created if the flow log creator can create and modify bucket policies.
    - Security groups
        - What traffic can flow to/from an instance
        - Allow rules only, direction specific.
        - Multiple SGs per instance are possible.
        - Rules on src/dest, dest port, protocol (TCP, UDP, etc)
        - src/dest can be ip range; a sg in this VPC or a peered one; service prefix list for gateway endpoints
        - Default rules in a new group: no inbound, all outbound.
        - The default security group also allows inbound from other instances in the sg.
        - Stateful - responses are always allowed
        - Can reference SGs in peered VPCs.

Storage

- **S3**
    - Monitoring
        - CloudTrail by default records bucket-level actions
        - Can enble CloudTrail logging of object-level actions by setting that property on a bucket in S3 (can choose read/write)
        - Server access logging - separate audit log, configured per-bucket, that stores events in a bucket. Destination bucket needs a special ACL (see ACL section). Best-effort delivery.
    - Buckets and Objects are the main resources, each have various subresources (versioning, policies/acls, ...)
    - Buckets are truly global - no region or account ID in their ARN
    - The account that uploads objects owns them - even if the bucket is owned by a different account! Bucket owner pays for storage, manages storage class, and can delete or deny access to any object.

- Access control logic is complex. That page doesn't include "block public access" logic.
  - User needs to have permission - using identity policies (or user is the root of an account)
  - For bucket operations: bucket needs to have permission - either just bucket policy/acl for user in a different account, or both bucket policy/acl and identity policy if user is in the same account
  - For object operations: User has to have permission (or be root). Bucket policy/acl has to *not deny*. Object ACL (or bucket policy) has to allow. Three different account contexts in play - the user's account (IAM), the bucket's account (for bucket ACL/policy & identity policy if same-account), the object's account (for object ACL).
- Bucket policies
  - Bucket resource-based policy.
- ACLs
  - Bucket and object resource-based policy
  - Default ACL grants the owner account full control
  - List of grants, each grant gives a grantee (an AWS account or predefined group) a permission
  - Grantee groups: Authenticated Users group - *any* AWS user. All Users group - incl anonymous. Log Delivery group - S3 audit logs.
  - Permissions: READ, WRITE (only applies to buckets - allows overwriting and deleting objects), READ/WRITE ACL, FULL CONTROL (all of the above)
  - Don't use bucket ACLs except for allowing write access to the Log Delivery Group for access logging. This is the only way.
- Block Public Access
  - Applied to specific buckets, or all buckets in an account
  - BlockPublicAcls - can't create new public bucket or object ACLs
  - IgnorePublicAcls - existing (and new) public ACLs are ignored
  - BlockPublicPolicy - can't create public bucket polciies (only really works if applied account-wide, otherwise you can undo it via a bucket policy that allows modifying this policy...)
  - RestrictPublicBuckets - blocks all anonymous and cross-account access to a bucket
- Query string authentication - instead of using the authorization header, you specify the access key ID and signature in
- Event notifications

        - Per bucket.
        - Sources: object creation, deletion, restoration from Glacier, and loss (for reduced redunadancy class)
        - Destinations: SNS topic, SQS queue, Lambda
    - Versioning
        - Enable on a bucket, then all object versions (including deleted one) remain available. Bucket owner can permanently delete.
        - Object lock: can't be deleted or overwritten until a particular date. Governance mode - needs s3:BypassGovernanceMode to override; Compliance mode - can't be overridden, even by root. Legal Hold - no end date (separate perm needed to override). Applies to an individual object version.
        - MFA delete: have to provide a TOTP code to delete (separate to IAM MFA) in `x-amz-mfa` header
    - Lifecycle policies
        - Transition action - change storage class
        - Expiration action - delete
        - e.g. archive old versions to glacier, then delete.
    - Encryption
        - SSE-S3 - pure S3 managed encryption
        - SSE-KMS - standard KMS integration like other services
        - SSE-C - you send the plaintext encryption key in the request (!)
        - The SDKs also ease support for client-side encryption

- Elastic Block Store (EBS)

    - Redundancy but only within a single AZ
    - Snapshots might be useful for recovery
    - Encryption (if enabled) happens on the EC2 server side (outside the EC2 VM), hence encrypted in transit and rest. Uses KMS - wrapped data key stored alongside volume.
    - `ec2:CreateVolume` action paired with `ec2:Encrypted` condition key can enforce use of encrypted volumes

- EFS

    - NFS filesystem
    - Standard posix permissions
    - Mount targets appear as endpoints in a VPC, so Security Groups can control access
    - IAM only used for administration

  - transparent encryption at rest with KMS (could monitor compliance with a CloudWatch alarm over CloudTrail logs)
  - NFS over TLS is an option with the EFS mount helper (stunnel)

- S3 Glacier
  - Encrypted by default
  - Value access policies - resource based policy attached to a vault. Like a bucket policy.
  - Vault lock policies - a vault access policy that can be locked to prevent changes to it
  - Other than the global ones and tags, supports `glacier:ArchiveAgeInDays` condition key - nice in combo with the `glacier:DeleteArchive` action
  - Retrieval requires job initiation then getting the output from the job
  - Data retrieval policy: a resource-based policy for regions? They don't describe it as such, but each region can have one policy that constrains Glacier retrievals to free tier / maximum transfer rate / unlimited.

- Backup
  - Centralise backups across RDS, DynamoDB, EBS, EFS, Storage Gateway. Uses those services' native capabilities (snapshots etc)
  - Can be encrypted in transit and at rest. Uses the service's native encryption capabilities, or for EFS where the backup functionality comes from Backup itself, it does the usual KMS encryption. Other than EFS, encryption depends on whether the source is encrypted (note DynamoDB tables are always encrypted at rest).
  - Resources: plans, vaults, recovery points.
  - Resource-based policy for vaults, but these only constrain *vault* access, not access to the underlying backup like an EBS or RDS snapshot.

- Snow family
  - All use encryption integrated with KMS. Encryption is performed client-side prior to transfer to the device.
  - Snowball and Snowball edge use tamper-resistant designs and active monitoring using a TPM
  - API calls use IAM as normal. The Snowball devices don't - combo of an encrypted manifest & access code give full control of it.
  - Snowmobile is a little different :D ... "dedicated security personnel, GPS tracking, alarm monitoring, 24/7 video surveillance, and an optional escort security vehicle"

- Storage Gateway
  - SMB/NFS front end to S3 - file gateway
  - iSCSI front end to Glacier/S3 - tape gateway / volume gateway
  - Encrypted in transit and at rest. By default uses SSE-S3, can configure to use SSE-KMS.
  - iSCSI has its own authentication model (CHAP)

In-depth Chapter Wise review

AWS Security 101

Security Basics

CIA

- **Confidentiality**: IAM, MFA, bucket policies, security groups, ACL's within VPC's, KMS encryption etc.
- **Integrity**: Certificate Manager (SSL), IAM, bucket policies, S3 version control, MFA for S3 deletion etc.
- **Availability**: Autoscaling, Multi-AZs, Route53 health checks etc.

AAA

- **Authentication**: auth into IAM entity (user/role)
- **Authorization**: IAM policies to define access
- **Accounting**: audit trail i.e. CloudTrail

Non-repudiation

- Not being able to deny something has happened.
- CloudTrail, CloudWatch.

How does AWS Secure their Stuff?

Physical and Environmental Security

- AWS consist of regions, with 2+ availability zones, each made up of multiple data centres.

- Secured by fire-detection/suppression, power (2 feeds / different power sources), climate and temperature, management (ex-soldiers / physical access in-out), storage device decommissioning (zero out all data from disks and then shredding/smashing disk).

Business Continuity Management

- Monitor availability, incident response
- Perform company-wide executive reviews when an incident has occurred + communicating issues out to customers

Network Security

- Secure network architecture
- Secure access points - everything is available as a public API, so access points but be secured.
- Transmission protection e.g. TLS security on console login, S3 bucket access etc.
- Amazon corporate segregation - Amazon.com network is completely different network to AWS. Bastion is required for employee access from Amazon.com -> AWS
- Fault-tolerant design - multiple AZ's in multiple regions
- Network monitoring and protection - DDoS mitigation via. AWS shield + advanced DDoS mitigation is available (more costly)

AWS Access

- Account Review and Audit - AWS users' accounts are audited/reviewed every 90days. If account has not been used, it will be revoked and reapplying for access is required.
- Background Checks
- Credentials Policy - Amazon's password policies (very complex, changed every 90 days).

Secure Design Principles

- Formal design reviews by AWS security team, threat modelling, completion of risk assessments, static code analysis tools run as part of build process, all deployed software undergoes re-occurring penetration testing prepared by carefully selected industry experts.

Change Management

- AWS performs routine emergency and configuration changes to AWS infra.
- It is all authorized, tested, logged, approved and documented in accordance with industry norms for similar systems.
- AWS communicates to customers via. email / service health dashboard.

Why should we trust AWS?

- AWS meets a whole bunch of compliance programs / IT security standards.
- Big ones are ISO27001, PCIDSS compliant, HIPAA (medical records).
- Your own software/infrastructure requires a GAP-AUDIT.

Exam Tips

- Remember different security controls around: physical and environmental security, business continuity, network security, AWS access, secure design principles, change management.
- Remember that the corporate Amazon.com network is completely segregated from the AWS network. Permissions / reviews are required when an employee wants to access AWS. Permissions are revoked as soon as nologin for 90 days.

Shared Responsibility Model

What is it?

- Security WITHIN the cloud is the responsibility of the customer.
- E.g. House example:
    - Landlord is responsible for installing fire alarms, fences.
    - You are responsible for locking your door, making sure windows are shut etc.

AWS Security Responsibilities

- Global infrastructure - their data centres
- Hardware, software, networking and facilities - all their hardware, software such as RDS / AWS operation systems etc.
- Managed services - S3, DynamoDB etc.

Customer Security Responsibilities

- Infrastructure as a Service
- Updates and security patches e.g. EC2 instances etc.
- Configuration of AWS-provided firewall - VPC rules, security groups, network ACLs etc.

Diagram of the Shared Responsibility Model: https://aws.amazon.com/compliance/shared-responsibility-model/

Basically, if the customer has no access to the underlying OS/software/infrastructure, then it is AWS's responsibility.

Shared Responsibility - AWS service types

Infrastructure services - compute services such as EC2, EBS, Auto Scaling, VPC

- You can architect and build cloud infrastructure, control the OS, configure and operate any identity management system that provides access to the user layer of the virtualization stack.
- EC2 - AMIs, OS, applications, data in transit, data at rest, data stores, credentials, policies and configuration.

Container services - services such as RDS, Elastic Map Reduce (EMR) and Elastic Beanstalk.

- RDS example - you have a DB that you can install/access but you don't manage the underlying OS. AWS is responsible for patching for the RDS instance.
- Services that are typically run on separate EC2s or other infrastructure instances. Sometimes you don't manage the OS or platform layer.
- Customer is responsible for setting up and managing network controls such as firewall rules, managing platform-level identity and access management separately from IAM.

Abstracted services - services such as S3, Glacier, DynamoDB, SQS, SES.

- Services that abstract the platform or management layer on which you can build and operate cloud applications.
- Customer can access the endpoints of these abstracted services using AWS APIs.
- AWS is responsible for managing the underlying service components or OS in which these services reside.

Exam Tips: Have a STRONG understanding of the shared responsibility model.

- The model changes for the three different service types:

1. Infrastructure services (EC2, EBS, VPC)
2. Container services (RDS, EMR, Elastic Beanstalk) - AWS responsible for the OS, container itself.
3. Abstracted services (S3, Glacier, DynamoDB, SQS, SES) - AWS responsible for almost everything, except for the application layer e.g. TLS / access controls.

Security IN AWS

Controls that you need:

- **Visibility**: AWS Config - managed and custom rules

- **Auditability**: AWS CloudTrail - records every API call in the environment
- **Controllability**:
    - AWS KMS - multi-tenant. Underlying hardware is shared, but strict controls.
    - AWS CloudHSM (hardware security module) - dedicated. Underlying hardware is NOT shared. **Exam: Which service is required for FIPS 140-2 Compliance? - CloudHSM as KMS being multi-tenant/shared does not comply.**
- **Agility**:
    - AWS CloudFormation - deploy templates to any regions
    - AWS Elastic Beanstalk - AWS provision resources for you, rather than you doing it each service manually
- **Automation**:
    - AWS OpsWorks - operate alongside CF / EB
    - AWS CodeDeploy - operate alongside CF / EB
- Scale__: Every customer gets the same AWS security foundations, from a startup to a Fortune 500 company.

Other services applying to all controls

- AWS IAM - creating users, password policies, MFA, groups
- AWS CloudWatch - monitor environment, see breaches, CPU runtime
- AWS Trusted Advisor - advises on security, budgeting, system performance and reliability

Identity Access Management, S3 & Security Policies

IAM provides:

- Centralised control of your AWS account
- Shared access to your AWS account
- Granular permissions
- Identity Federation (Active Directory, Facebook, Linkedin etc.)
- MFA
- Provide temporary access for users/devices and services where necessary.
- Allows you to set up your own password rotation policy
- Integrates with many AWS services
- Supports PCI DSS Compliance

Critical Terms:

- Users - end users
- Groups - collection of users under one set of permissions (apply one policy to group, users inherit policy)
- Roles - assign roles to AWS resources (e.g. EC2, Lambdas)
- Policies - document that defines permissions

IAM is global - users, groups, roles, policies are done on a global level, not region-specific.

IAM Permissions Boundary for IAM Entities (users/roles)

- A Permissions Boundary is using a managed policy to set the *maximum permissions* that an identity-based policy can grant to an IAM entity.
- https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_boundaries.html
- https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_iam-condition-keys.html#ck_PermissionsBoundary

IAM Root User Scenario

Scenario: You have have started as a sysadmin at a cloud-based company. Previous admin used only the root-user.

First thing to do = rotate everything

- Change password
- De-activate then re-activate MFA
- Delete Access Key ID / Secret Access Key (don't create new access keys via. root user)
- Verify and delete IAM users that are not-legitimate.

IAM Policies

IAM policies specify what you are allowed to do with any AWS resource. You attach IAM policies to users, groups or roles.

Types of IAM policies:

- **AWS Managed Policies**: standalone policy managed by AWS. Managed policy `AWS Administrator` has access to IAM, whereas `AWS PowerUser` does not.
- **Customer Managed Policies**: standalone policy that you create and manage.
- **Inline Policies**: Used if you want to maintain a strict ONE-TO-ONE relationship between policy and the principal entity that its applied to.

S3 Bucket Policies

S3 bucket policies specify what actions are allowed or denied on the bucket.

- They are attached only to S3 buckets.
- They are BUCKET-LEVEL only (not bucket object-level).

Why use S3 policy instead of IAM policy

- You want to grant cross-account access to your S3 environment, without using IAM roles.
- Your IAM policies reach the size limit (2kb for users, 5kb for groups, 10kb for roles). S3 supports bucket policies of up to 20kb.
- You prefer to keep access control policies in the S3 environment.

Management of individual S3 buckets = best use case of S3 bucket policies.

- Having a deny policy for a specific bucket is easier than creating an IAM policy that denies access to a specific bucket, then rolling that out to every user in your organisation.
- Example scenario: bucket could contain everyone's performance reviews in it.

Use the **AWS Policy Generator** to generate a S3 bucket policy.

In any AWS policy (IAM, S3, Key), a **DENY will always override an ALLOW**.

S3 Object Access Control Lists (ACLs)

S3 ACLs are a legacy access control mechanism. AWS recommends sticking to IAM policies and S3 bucket policies. However, S3 ACLs can be applied to individual objects/files as opposed to S3 bucket policies.

S3 ACL use cases:

- If you need **fine grained permissions** on individual files/objects.
- Reaching **size limit of 20kb** for S3 bucket policies.

Managing S3 object permissions

- Click on object itself -> permissions
- Applying S3 object policies to individual IAM users - possible but can only be done via. CLI or AWS API (not console).
- Add S3 object access for other AWS Accounts by adding Account ID.

Conflict policy example: IAM user policy denying all S3 read vs. S3 bucket with object open to the public.

- Even though an explicit DENY overrides all ALLOW policies… the user would still be able to access the object. WHY??? =>
- The user CAN access objects in the public bucket via. the public bucket link (as an anonymous user).
- The user CANNOT access objects in the public bucket via. opening the object within AWS console/CLI/API (as an AWS user).

EXAM: Best exam practise is by creating your own S3 Bucket Policies, S3 Object ACLs, IAM User Policies etc.

Policy Conflicts (EXAM ESSENTIAL TOPIC)

What happens if an IAM policy conflicts with an S3 policy which conflicts with an S3 ACL?

Whenever an AWS principal (user, group or role) issues a request to S3, the authorization decision depends on the union of all the IAM policies, S3 bucket policies and S3 ACLs that apply.

Least-privilege:

- Decisions ALWAYS default to DENY.
- An explicit DENY ALWAYS trumps an ALLOW.
- So if you DENY access to something somewhere and then something else allows access the DENY will override the ALLOW.

ACCESS DENIED EXAMPLES:

- IAM policy grants access to an object + S3 bucket policy denies access to object + no S3 ACL exists.
- No method specifies an ALLOW, request is denied by default.

ACCESS ALLOWED EXAMPLE:

- No method specifies a DENY + one or more methods specify an ALLOW.

Policy Conflict flow:

1. Decision starts at DENY by default.
2. Any applicable policies? (YES = CONTINUE | NO = DENY)
3. Does a policy have an EXPLICIT DENY? (YES = DENY | NO = CONTINUE)

4. Does a policy have an ALLOW? (YES = ALLOW | NO = DENY)

This flow will be examined heavily with scenarios containing 2-3 different policies.

Forcing Encryption on S3

Use S3 bucket policy to enforce encryption - prevent read without SSL enabled:

```
// If secure transport is false, DENY read.
// Alternative policy, if secure transport is true, ALLOW read.
"Sid":"PublicReadGetObject",
"Effect":"Deny",
"Principal":{
    "AWS":"*"
}
"Action":"s3:GetObject",
"Resource":"arn:aws:s3:::bucketname/*",
"Condition":{
    "Bool":{
        "aws:SecureTransport":false
    }
}
```

Cross-Region Replication

Cross-region replication replicates objects from one region to another. By default, this is done using SSL. You don't need to enable encryption.

You can replicate objects from a source bucket to only one destination bucket (1-1 relationship). After S3 replicates an object, the object can't be replicated again.

Cross-Region Replication (CRR) requirements:

- Src/dest buckets must have **versioning enabled**.
- Src/dest buckets must be in **different AWS regions**.
- Amazon S3 must have permissions to replicate objects from src/dest bucket on your behalf. When you enable CRR for the first time, a **role will be created** for you + a **customer-managed policy** will be assigned.
- If src bucket owner also owns the object, the bucket owner has full permissions to replicate the object. If not, object owner must grant the bucket owner `READ`/`READ_ACP` permissions via. the object ACL.

CRR + Cross Accounts:

- The IAM role must have permissions to replicate objects in the destination bucket.
- In CRR config, you can optionally direct AWS S3 to change ownership of object replicas to the AWS account that owns the destination bucket.
- **AUDIT account Cross-Region Replication use case**:

1. CloudTrail logs AWS accounts XYZ.
2. Turn on CRR to replicate logs to AUDIT account.
3. AWS accounts XYZ can only replicate logs, but not read/write logs in AUDIT account.

What is replicated?

- New objects created after you add a replication config.
- S3 replicates objects encrypted using S3 managed keys (SSE-S3) or KMS managed keys (SSE-KMS) + unencrypted objects.
- Object metadata
- Object ACL updates
- Object tags
- S3 replicates only objects in the src bucket for which the bucket owner has permissions to read objects and read access control lists.

DELETE marker replication

- Delete markers on an object are replicated. Deletion of versions of objects are NOT replicated.
- A delete marker only hides an object via. versioning, not actually delete it.

What is NOT replicated

- Anything created **BEFORE CRR is turned on**.
- Objects created with SSE using **customer-provided (SSE-C)** encryption keys.
- Objects created with SSE using **AWS KMS-managed encryption (SSE-KMS)** keys, unless you explicitly enable this option.
- Objects in the src bucket for which the **bucket owner does NOT have permissions** (happens when the obj owner is different from the bucket owner).
- Delete replication of a particular VERSION of an object. Source bucket deletion != dest bucket deletion. This is to stop malicious deletion of a specific version on an object.

Resources:

- Cross-Region Replication: https://docs.aws.amazon.com/AmazonS3/latest/dev/replication.html
- What does S3 replicate: https://docs.aws.amazon.com/AmazonS3/latest/dev/replication-what-is-isnot-replicated.html

Securing S3 Using CloudFront

Force users to only access S3 via. CloudFront instead of direct access via. S3 URL.

Steps to create a new CF distribution:

1. CloudFront service
2. Create a new distribution -> Web Distribution
3. Origin Domain Name: the source S3 bucket URL
4. Restrict Bucket Access -> NO (exam will test how to restrict AFTER a distribution has already been created)
5. Everything as default

Steps to secure S3 bucket via. CF:

1. Goto CloudFront -> **Origins and Origin Groups**
2. Turn on **Restrict Bucket Access** -> Create an **Origin Access Identity**
3. Turn on **Grant Read Permissions on Bucket** to allow CloudFront OAI to perform `s3:GetObject`

S3 bucket policy to restrict access via. CloudFront:

```
{
        "Sid": "1",
        "Effect": "Allow",
        "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
Identity EAF5XXXXXXXXX"
                },
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET/*"
}
```

Using SSL Certificates using CloudFront

DEFAULT SSL CERTIFICATE: If you are happy for users to access your content using *.cloudfront.net domain name.

CUSTOM SSL CERTIFICATE: If you want to use a domain name that you own example.com.

You must store your custom SSL Certificate using:

- IAM API
- AWS Certificate Manager (ACM)
- Only in the `us-east-1` region = US East (N. Virginia)

Secure S3 Using Pre-Signed URLs

Another method of accessing objects inside S3 - done via. SDKs (Python, Java, Go) or CLI.

```
$ aws s3 mb s3://acloudgurupresigned                # Make bucket
$ echo "Hello Cloud Gurus" > hello.txt
$ aws s3 cp hello.txt s3://acloudgurupresigned      # Upload object to bucket
$ aws s3 ls s3://acloudgurupresigned                # Check object is in bucket
$ aws s3 presign s3://acloudgurupresigned/hello.txt --expires-in 300 # presign URL
with 300 sec expiration (default expiry = 1 hr)
https://acloudgurupresigned.s3.amazonaws.com/hello.txt?AWSACcessKeyId=XXX&Expires=
XXX&x-amz-security-token=XXX&Signature=XXX
```

Security Token Service (STS) (IMPORTANT EXAM TOPIC)

STS grants users limited and temporary access to AWS resources.

These users can come from:

- Federation (typically Active Directory)
  - Uses SAML
  - Grants temp access based off user's AD credentials. Does not need to be a user in IAM.
  - SSO allows users to log into AWS console without assigning IAM credentials.
- Federation with Mobile Apps
  - Facebook / Amazon / Google or other OpenID providers.
- Cross Account Access
  - Lets users from one AWS account access resources in another.

Key Terms:

- Federation - combining or joining a list of users in one domain (such as IAM) with a list of users in another domain (such as AD, Facebook etc.)
- Identity Broker - service that allows you to take an identity from point A and join it (federate it) to point B.
- Identity Store - services like AD, Facebook, Google etc.
- Identities - a user of a service like Facebook etc.

An Identity is a user, that is stored in an Identity Store (like Active Directory/Facebook). You create an Identity broker that allows you take those Identities in your Identity Store and join them up to IAM -> This is essentially the federation/joining of IAM with AD/Facebook. The service that allows this is the Security Token Service.

Scenario: *You are hosting a company website on EC2 web servers in your VPC. Users of the site must login to the site, which authenticates against the company's AD servers which are based on-site at the company HQ. Your VPC is connected to the company HQ via. a secure IPSEC VPN. Once logged in, the user can only have access to their own S3 bucket.*

How to set this up:

1. Develop an Identity Broker (join AD -> IAM).
2. Identity Broker will authenticate (using client/appid, secret) against AD:
    - Authenticate to obtain an AD token.
    - Pass AD token to STS.
    - STS will provide us with another token.
3. Pass STS to the web application to authenticate against S3.
4. S3 uses IAM to check if user has access to S3.
5. User is able to access S3.

Scenario:

1. Employee enters username / password
2. Application calls Identity Broker. Broker captures username/password.
3. Identity Broker uses the organisation's LDAP directory to validate the employee's identity.
4. Identity Broker calls the `sts:GetFederationToken` API using IAM credentials.
    - GetFederationToken(DurationSeconds, Name, Policy, PolicyArn) where:
    - *DurationSeconds*: duration of the STS token (1 to 36 hours).
    - *Name*: name of the federated user.
    - *Policy*: inline IAM policy.
    - *PolicyArn*: ARN referencing an IAM policy.
5. STS confirms that the policy of the IAM user making the call to GetFederationToken gives permission to create new tokens.
6. STS returns the temp STS token to the Identity Broker.
7. Identity Broker returns the STS token to the application.
8. Application uses the STS token to make requests to S3.
9. S3 uses IAM to verify STS token and to allow requested operation on the given S3 bucket.
10. IAM provides S3 with go-ahead to perform requested operation.

High-Level Summary:

1. Authenticate (as Identity/User) against 3rd-party (Identity Store: AD/Facebook/Google).
2. Authenticate (as Identity Broker) against STS.
3. Authenticate (as Application) against AWS service to obtain access to resource.

Web Identity Federation / Amazon Cognito

Web Identity Federation lets you give users access to AWS resources after they have successfully authenticated with a web-based identity provider like Amazon/Facebook/Google. User trades authentication code from Web ID provider for an AWS STS token.

Suggested use case: mobile app which you want to make available to Facebook users. (recommended for social accounts)

Amazon Cognito

- Sign-up / Sign-in to your apps
- Provides guest access
- Acts as identity broker between your app / Web ID provider
- Synchronises user data across multiple devices (mobile, desktop data sync)
- Recommended for mobile apps running on AWS.

Amazon Cognito scenario:

- Mobile shopping app: S3 for product data, DynamoDB for customer data.
- User logs into Facebook, Facebook provides web token.
- Cognito takes web token and exchanges it for STS token.
- Cognito passes STS token to mobile app.
- Mobile app uses STS token to get access to resources for user.

Amazon Cognito benefits:

- No need for mobile app to embed or store AWS credentials locally on the device = increased security.
- Provides users a seamless experience across all devices.

Cognito User Pools: user directories used to manage sign-up and sign-in functionality for mobile/web apps.

- User sign-in directly via. User Pool or indirectly via. identity provider (Amazon/Facebook/Google)
- Cognito acts as identity broker between ID provider and AWS.

Glacier Vault Lock

Glacier is a low-cost storage service for data archiving and long-term backup.

- *Archives*: a single file or multiple files stored in a .tar or .zip.
- *Vault*: containers which store one or more Archives

- *Vault Lock Policy*: similar to an IAM policy to configure and enforce compliance controls - configure write-once-read-many archives / create data retention policies

Example Vault Lock Policy: Enforce archive retention for 1 year (deny archive delete for all archives <365 days old)

```
"Version":"2012-10-17",
"Statement":[
    {
        "Sid":"deny-based-on-archive-age",
        "Principal":"*",
        "Effect":"Deny",
        "Action":"glacier:DeleteArchive",
        "Resource":[
            "arn:aws:glacier:us-west-2:XXXaccountidXXX:vaults/examplevault"
        ],
        "Condition":{
            "NumericLessThan":{
                "glacier:ArchiveAgeInDays":"365",
            }
        }
    }
]
```

Steps to configuring Vault Locks:

- Create Vault Lock policy.
- Initiate lock by attaching Vault Lock policy to your vault = in-progress state.
- You have 24 hours to validate the lock policy. You can abort within 24 hours.
- Once validated, Vault Lock policies are immutable.

Vault Lock Policy vs. Vault Access Policy:

- https://docs.aws.amazon.com/amazonglacier/latest/dev/vault-lock.html

AWS Organisations

AWS Organisations is an account management service that lets you consolidate multiple AWS accounts into an organisation so that you can consolidate billing, group your AWS accounts into logical groupings for access control and attach Service Control Policies.

SCPs enable you to restrict, at the account level of granularity, what services and actions the users, groups, and roles in those accounts can do. However, an SCP never grants permissions. The SCP limits permissions for entities in member accounts, including each AWS account root user. SCPs are available only in an AWS organization that has all features enabled, SCPs aren't available if your organization has enabled only the consolidated billing features.

https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scp.html

IAM Credential Report

IAM Credential Report is a CSV-formatted report which lists all users in the accounts + status of their various credentials, including

- Passwords: enabled, last used, last rotated, next rotation.
- Access Keys: similar to above + last used.
- MFA devices: similar to above.

CLI commands for IAM Credential Reports

```
# generate a credential report
aws iam generate-credential-report
# download a credential report / same API call but base64 decode
aws iam get-credential-report
aws iam get-credential-report --output text --query Content | base64 -D
```

Required permissions to generate IAM Credential Reports

- `GenerateCredentialReport`: create report
- `GetCredentialReport`: download report

An IAM Policy with permissions to generate IAM Credential Reports

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "VisualEditor0",
        "Effect": "Allow",
        "Action": [
            "iam:GenerateCredentialReport"
        ],
        "Resource": "*"
    }]
}
```

Summary / Exam Tips

Resetting Root Users

- Create new root user password / strong password policy.
- Delete 2FA / re-create.
- Delete Access Key ID / Secret Access Key.
- Check existing user accounts, delete if not legit.

IAM policies

- IAM is Global.
- Three different types: (1) Managed Policies (2) Customer Managed Policies (3) Inline Policies

S3 policies

- S3 policies are attached only to S3 buckets (NOT objects). They specify what is ALLOWED/DENIED on the bucket.
- Broken down to the user-level.
- *EXPLICIT DENY ALWAYS OVERRIDES AN ALLOW.*
- S3 ACL's: Legacy access control for enforcing access to S3 OBJECTS.
- S3 policy conflicts: see *policy conflict diagram* above (IMPORTANT).
- aws:SecureTransport: restrict S3 bucket access to only HTTPS.
- Cross-Region-Replication (CRR):
    - Delete markers are replicated, deleted versions of files are NOT replicated.
    - Versioning must be enabled.
    - Possible to use CRR from one AWS account to another
    - SSL is enabled by default when you configure CRR
    - IAM role must have permissions to replicate objects in destination bucket.
    - Scenario: replicate CloudTrail logs to separate AWS audit account (can only send data there, not read/write).

Pre-signed URLs (CLI/SDK only):

- Access objects using pre-signed URL's
- Exist only for a certain length of time.
- Change TTL by using `expires-in`

STS / Identity Provider

- User provides credentials to Identity Provider (AD/FB/Google) -> AWS STS -> User accesses AWS resource -> AWS resource checks IAM -> access is provided to user.

Logging and Monitoring

AWS CloudTrail

AWS CloudTrail is a web service that records AWS API calls for your account and delivers log files to you.

- User interacts with AWS platform via. Console or API Call.
- CloudTrail logs all these interactions with AWS services (only API calls).
- CloudTrail will NOT log actions such as SSH/RDP into an EC2.

Enables:

- After-the-fact incident investigation
- Near-realtime intrusion detection
- Industry and regulatory compliance

Provides:

- Logs API call details (for supported services)

Supported services: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-aws-service-specific-topics.html Un-supported services: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-unsupported-aws-services.html CloudTrail limits: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/WhatIsCloudTrail-Limits.html

Log info:

- Metadata around API calls
- Identity of API caller
- Time of API call
- Source IP of API caller
- Request params
- Response returned by the service

Where? CloudTrail Event Logs:

- Sent to an S3 bucket
- You manage retention in S3
- Delivered every 5 minutes to S3 with up to 15 minute delay
- SNS notifications available - e.g. notify you if something happens
- Can aggregate across multiple regions
- Can aggregate across multiple accounts - good for non-repudiation. Bad actor can only destroy within account, not audit account.

Setup:

- CT enabled by default (only keeps 7-day audit trail), you will need to provision to have it for longer.
- Management Events: ALL READ/WRITE
- Data Events (s3 object activity): leave as default = not enabled
- Storage Location: create a new S3 bucket

Validating CT log file integrity:

- SHA-256 hash
- SHA-256 with RSA for digital signing
- Log files are delivered with a 'digest' file
- Digest files can be used to validate the integrity of the log file
- You can use the AWS CLI to perform validation.

CloudTrail Log Protection

Log files are encrypted by default (AES-256) even if the bucket itself doesn't show encryption turned on.

CT logs must be secured because they contain valuable info to an attacker such as:

- Personally identifiable info such as usernames / team membership.
- Config information such as a DynamoDB table and key names may be stored.

How to stop unauthorised access?

- Use IAM policies
- Use S3 bucket policies to restrict access
- Use SSE-S3 or SSE-KMS to encrypt logs

How do we restrict access to only employees with a security responsibility?

- Place employees who have a security role into an IAM group with attached policies
- Two AWS-managed policies: AWSCloudTrailFullAccess (security role) and AWSCloudTrailReadOnly (auditor role)

How can we be notified that a log file has been created / validate integrity?

- Configure SNS notifications and log file validation.
- Develop a solution to execute log validation using the digest file.

How to prevent CT log files from being deleted?

- Using IAM and bucket policies
- Configure S3 MFA delete
- Validate that logs have not been deleted using log file validation

How to ensure that logs are retained for X years?

- By default, logs are kept indefinitely
- Can use S3 Object Lifecycle Management to delete files after required period of time.
    - Go to S3 bucket -> Management Tab -> "Add lifecycle rule" button -> Configure bucket expiration
- OR move files to AWS Glacier for long-term storage.

AWS CloudWatch

AWS CloudWatch is a monitoring service for AWS cloud resources and the applications you run on AWS.

Enables:

- Resource utilisation,. operational performance monitoring
- Log aggregation and basic analysis

Provides:

- Real-time monitoring within AWS for resources and applications
- Hooks to event triggers

Key components:

1. CloudWatch
2. CloudWatch Logs
3. CloudWatch Events

CloudWatch:

- Real-time monitoring: standard monitoring (every 5 min) / detailed monitoring (every 1 min)
- Metrics: CPU utilisation, network utilisation
- Alarms: CPU > 80%, trigger alarm

- Notifications: SNS notifications etc.
- Custom Metrics: pass / program custom metrics via. AWS API.

CloudWatch Logs:

- Pushed from some AWS services, including CloudTrail
- Pushed from your application/systems - kernel logs, application logs, web-server logs etc.
- Metrics from log entry matches
- Stored indefinitely (not user S3)

CloudWatch Events | scenario: user creating EC2 instance, resulting in auto-deletion via. CloudWatch Events

1. User performs API call (create EC2)
2. API call logged in CloudTrail S3 bucket
3. CloudTrail is configured as a CloudWatch Event Source, so API call is pushed to CloudWatch Events
4. CloudWatch Events pushes details of API call to an Event Target, such as an AWS Lambda
5. AWS Lambda deletes EC2 instance.

CloudWatch Events:

- Near real-time stream of system events
- Events:
  - AWS Resources state change
  - AWS CloudTrail (API Calls)
  - Custom events (e.g. HTTP 403 status in Apache web-server logs)
  - Scheduled events
- Rules: match incoming events and route them to one or more targets
- Targets: Lambda, SNS topics. SQS queues, Kinesis Streams and more

AWS Config

AWS Config is a fully managed service that provides you with an AWS resource inventory, configuration history and configuration change notifications to enable security and governance.

Enables: Compliance auditing, security analysis, resource tracking (what resource we're using where) Provides: Configuration snapshots and log config changes of AWS resources, automated compliance checking

AWS Config needs to be deployed in each individual region. It doesn't automatically deploy in every region in your account.

How does it work:

1. AWS resource configuration change -> event fires off
2. AWS Config picks up event -> AWS Config logs event in S3 bucket
3. Event target = Lambda is triggered -> Managed or Custom rules (Lambda functions)
4. AWS Config will evaluate if configuration change has broken a rule
5. If rule is broken, AWS Config will trigger SNS notification and is sent to user

Terminology:

- *Configuration Items*: point-in-time attributes of resource
- *Configuration Snapshots*: collection of config items
- *Configuration Stream*: stream of changed items
- *Configuration History*: collection of config items for a resource over time
- *Configuration Recorder*: the configuration of AWS Config that records and stores config items (Config Recorder Role)

Recorder Setup:

- Logs config for account in region (per-region-basis)
- Stores in S3
- Notified of issues via. SNS

What we see:

- Resources Type, Resource ID
- Compliance checks:
    - Trigger:
        - periodic
        - configuration snapshot delivery (change in resource config -> trigger check)
    - Managed Rules: ~40 rules
- Timeline: configuration details, relationships, changes, CloudTrail events

Permissions needed for AWS Config - requires and IAM role with:

- ReadOnly permissions to the recorded resources
- Write access to S3 logging bucket
- Publish access to SNS

Restrict access to AWS Config:

- Users need to be authenticated with AWS and have appropriate permissions set via. IAM policies to gain access.
- Only Admins/Security needing to set up and manage Config require full access.
- Provide ReadOnly for Config day-to-day use e.g. analyse misconfigurations etc.

Monitoring Config:

- Use CloudTrail with Config to provide deeper insight into resources.
- Use CloudTrail to monitor access to Config - e.g. someone stopping Config Recorder would be monitored in CloudTrail.

AWS Config is a big part of the exam, so read the Config FAQ: https://aws.amazon.com/config/faq/

Set up an alert if Root user logs in / pro-active alerting (will be tested in exam)

Set up an alert if the Root user logs in and makes API calls

1. Turn on CloudTrail-CloudWatch logs integration
   - A role is required for CT to perform CloudWatch API calls. Two calls are performed:
   - `CreateLogStream`: Create a CloudWatch Logs log stream in the CloudWatch Logs log group you specify.
   - `PutLogEvents`: Deliver CloudTrail events to the CloudWatch Logs log stream.
2. Create a CloudWatch Metric Filter
3. Assign a metric
4. Create a Metric Alarm `{ $.userIdentity.type = "Root" && $.userIdentity.invokedBy NOT EXISTS && $.eventType != "AwsServiceEvent" }`
5. Test the alarm and receive an SNS notification
6. Look up the event and take corrective actions

AWS Cloud Hardware Security Module (CloudHSM)

This topic is not really examined - can mostly skip it.

AWS CloudHSM service helps meet corporate, contractual and regulatory compliance requirements for data security by using dedicated Hardware Security Module appliances within the AWS Cloud.

Enables: Control of data, evidence of control, meet tough compliance controls
Provides: Secure key storage (generate, store public/private keys), cryptographic operations, tamper-resistant Hardware Security Module

AWS Inspector and AWS Trusted Advisor - examined

AWS Inspector

- Automated security assessment service that helps improve security/compliance of applications on AWS.
- After performing an assessment, AWS Inspector produces a detailed list of security findings prioritised by level of security.
- Findings can be reviewed directly or as part of a report available via. AWS Inspector or API.
- How does it work (scenario: assessment target is an EC2/prod-webserver)
    i. Create an assessment target
    ii. Install agents on EC2 instances
    iii. Create "Assessment Template"
    iv. Perform an "Assessment Run"
    v. Review "Findings" against "Rules"
- Master template: Testing all rules - multiple rules packages over a 24 hour period
- Rule Packages: CVE's, CIS OS Config Benchmarks, Security Best Practices, Runtime Behaviour Analysis

AWS Trusted Advisor

- A service to advise you on Cost Optimisation, Performance, Security, Fault Tolerance.
    - *Basic Trusted Advisor*: Core checks and recommendations
    - *Full Trusted Advisor*: Business and Enterprise Companies only
- Some recommendations available to basic plan:

    - Security Groups (unrestricted ports), IAM use, MFA on Root, Service Limits (usage limits), exposed EBS snapshots etc.

Logging

Understand the 4 logging services and their differences: *CloudTrail, CloudWatch, Config, VPC Flow Logs*

Resources: White-paper *Security at Scale: Logging in AWS* https://d1.awsstatic.com/whitepapers/compliance/AWS_Security_at_Scale_Logging_in_AWS_Whitepaper.pdf

Control access to log files:

- Prevent unauthorised access (Authentication):
    - IAM users, groups, roles and policies
    - S3 bucket policies
    - MFA (IAM and S3 bucket policy level)
- Ensure role-based access (Authorization):
    - IAM users, groups, roles and policies
    - S3 bucket policies
- Alerts when logs are created or fail:
    - CloudTrail notifications
    - AWS Config rules
- Alerts are specific, but don't divulge detail:
    - CloudTrail SNS notifications only point to log file location, not show actual details.
- Log changes to system components:
    - AWS Config rules
    - CloudTrail
- Controls to prevent modification to logs:
    - IAM and S3 controls and policies
    - CloudTrail log file validation
    - CloudTrail log file encryption

Storage of log files:

- Logs are stored for at least 1 year
    - Store logs for an organisational-defined period of time
    - Store logs in real-time for resiliency
- S3

    - S3 Object Lifecycle Management
    - 99.99999% durability and 99.99% availability of objects over a given year

Infrastructure Security

AWS Key Management Service (KMS)

KMS is a managed service that makes it easy for you to create and control the encryption keys used to encrypt your data + uses Hardware Security Modules (HSMs) to protect the security of your keys.

KMS is region-specific.

Customer-Master-Keys (CMK)

- Is a logical representation of a master key, typically used to generate/encrypt/decrypt *Data Keys* used to encrypt your actual data - this practice is known as *Envelope Encryption*.
- CMKs consist of:
    - Alias
    - Creation date
    - Description
    - Key state
    - Key material (either customer provided or AWS provided)
- CMKs can NEVER be exported.
- You cannot delete CMKs immediately, only disable them with a 7-30 day waiting period before deletion.
- There are three types of CMKs:
    i. Customer managed CMKs - customer owned / imported keys in your account (full control)
    ii. AWS managed CMKs - AWS managed keys in your account that are associated with an AWS service
    iii. AWS owned CMKs - AWS owned keys that are NOT in your account for securing data in multiple AWS accounts (no control)

Customer-managed CMK: Importing your own Key Material into KMS

1. Create a customer-managed CMK with no key material by selecting "External" for the key material origin (not usable yet).
2. Import key material - select Wrapping Algorithm SHA1.
3. Import key material - download Wrapping Key (public key) as `PublicKey.bin` and Import Token `ImportTokenxxx`.
4. Use `openssl` and follow instructions here to generate key material and encrypt it with the Wrapping Key: https://docs.aws.amazon.com/kms/latest/developerguide/importing-keys-encrypt-key-material.html.
    - Generate a 256-bit symmetric key and save it in a file named `PlaintextKeyMaterial.bin`: `$ openssl rand -out PlaintextKeyMaterial.bin 32`
    - Encrypt the key material with the public Wrapping Key you downloaded earlier

```
5.  $ openssl rsautl -encrypt \
6.               -in PlaintextKeyMaterial.bin \
7.               -oaep \
8.               -inkey PublicKey.bin \
9.               -keyform DER \
10.              -pubin \
11.              -out EncryptedKeyMaterial.bin
```

12. Upload `EncryptedKeyMaterial.bin` and `ImportTokenxxx`.
13. The key is now available for use.

Why import your own Key Material:

- Compliance - prove that randomness meets your requirements.
- Extend your existing processes to AWS.
- Deletion of key-material without a 7-30 days wait.
- To be resilient to AWS failure by storing keys outside AWS.

Considerations of importing your own Key Material:

- You CANNOT use the same `EncryptedKeyMaterial` and `ImportToken` files twice - it is SINGLE USE only.
- You CANNOT *enable automatic key rotation* for a CMK with imported key material.
- You CAN *manually rotate* a CMK with imported key material - do this by creating a NEW CMK then import the new key material into that CMK (i.e. repeat the same process as creating a new key)
- You can delete imported keys immediately by deleting the Key Material.

Scenario #1: User disables a KMS key - event-driven security.

- User makes API call -> CloudTrail logs call -> CloudTrail sends Event Source to CloudWatch
- CloudWatch Event Rules is invoked -> Event Target for rule is a Lambda -> Lambda detects that user has disabled a key in KMS
- Lambda responds by auto re-enables key in KMS and/or fire off an SNS notification to security team.

Scenario #2: User disables a KMS key - AWS Config monitoring KMS events.

- AWS Config monitors and stores the KMS event into the Config S3 Bucket.
- Standard or Custom Rule (Lambda) is triggered which detects the KMS-disable.
- Rule will notify AWS Config -> AWS Config fires off SNS notification to security team.

Read the AWS KMS FAQ: https://aws.amazon.com/kms/faqs/

KMS Key Rotation Options

Extensive re-use of encryption keys is not recommended. Best practice is to rotate keys on a regular basis. Frequency of key rotation is dependant on local laws, regulations and corporate policies. Method of rotation depends on the type of key you are using.

1. AWS Managed Key
2. Customer Managed Key
3. Customer Managed w/ imported key material.

Key Rotation: AWS Managed Keys

- Automatic rotation every 3 years.
- No automatic rotation
- AWS manages everything and saves old backing key (key material)

Key Rotation: Customer Managed Keys

- Automatic rotation every 1 year (disabled by default)
- Manual rotation is possible
- Create a new CMK -> update apps / key-alias to use the new CMK (be careful of old-key deletion)

Key Rotation: Customer Managed Keys w/ Imported Key Material

- NO automatic rotation (key material is not generated in AWS)
- Manual rotation is the only option
- Create a new CMK -> update apps / key-alias to use the new CMK (be careful of old-key deletion)

Using KMS with EBS

Using KMS to encrypt Elastic Block Storage (EBS) volumes.

Creating an EBS encrypted volume w/ AWS-managed key:

1. Create a new EC2
2. Provision EBS storage (not encrypted by default)
3. Turn on encryption for the attached EBS volume.
4. This will generate an AWS-managed key for EBS in KMS.

- You cannot modify/delete this AWS-managed key.

How to encrypt an existing EBS volume / the Root Device volume (default vol when launching an EC2):

1. Create an EBS volume.
2. Create a snapshot of the EBS volume.
3. Create an Amazon Machine Image (AMI) from the EBS snapshot (actions -> create image).
4. Copy the AMI to a new image -> turn on encryption -> select either AWS-managed or your own CMK.
5. Launch the AMI. Your Root Device volume will now be encrypted.

EC2 and importing a Customer Managed Key Pair (for SSH access) - MAC USERS ONLY

1. Generate a private-key using RSA 2048 bits: `$ openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048`
2. Generate a public-key: `$ openssl rsa -pubout -in private_key.pem -out public_key.pem`
3. Change permissions of private-key: `$ chmod 400 private_key.pem`
4. Go to EC2 -> Key Pairs -> Import a Key Pair -> choose your public-key. Now you can provision an EC2 instance and select your public-key.

You CANNOT take your private/public-key pair and import it into KMS. You must follow the external Key Material import process to generate a CMK.

EC2 and Key Pairs (SSH access)

Creating additional/multiple key pairs for an EC2 instance.

1. Provision EC2 with an original key pair + SSH into instance `$ ssh ec2-user@public-ec2-ip -i KeyPairOriginal.pem`
2. Elevate to root `$ sudo su`
3. View your public keys by:
   - `$ cat ~/.ssh/authorized_keys` where authorized_keys contains all public keys.
   - OR by calling `$ curl http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key/`
4. Go to IAM -> create a new EC2 role -> provision `AmazonS3FullAccess` policy.
5. Go to EC2 -> attach new IAM role to instance.
6. Within the EC2, create a new S3 bucket: `$ aws s3 mb s3://brianec2keypairs`
7. Generate a new asymmetric key pair: `$ ssh-keygen -t rsa`
8. Add the new public key to authorized_keys `$ cat mynewkey.pub >> ~/.ssh/authorized_keys`.
9. Add the new private key to S3 bucket: `$ aws s3 cp mynewkey s3://brianec2keypairs`.
10. Go to S3 -> download new private key `mynewkey` -> `$ chmod 400 mynewkey`
11. Access the EC2 instance using the new private key `$ ssh ec2-user@ec2-public-ip -i mynewkey`

Notes about deleting Key Pairs:

- Deleting your key pair via. AWS Console will NOT prevent accessing EC2 with the private key, since the public key inside your EC2 in `~/.ssh/authorized_keys` still exists.
- If you delete an EC2 key pair via. AWS Console, you can generate a new key pair for the instance by:
  i. Go to the EC2 -> Actions -> Create an AMI.
  ii. Go to AMIs -> launch the EC2 clone -> create a new key pair.
  iii. Your new public key will be added to the existing list in `~/.ssh/authorized_keys`.
- Prevent access with old key pairs by removing the public keys in `~/.ssh/authorized_keys`.

Additional notes:

- You cannot use KMS with SSH for EC2 because Amazon is involved in generation of KMS keys.
- You can use CloudHSM with SSH for EC2 because you can export CloudHSM keys.

AWS Marketplace Security Products

You can purchase security products from 3rd-party vendors on the AWS Marketplace.

- Includes: firewalls, hardened OS's, WAF's, Antivirus, Security Monitoring etc.
- Billed: free, hourly, monthly, annually, BYOL etc.
- Recommended reading: steps on CIS OS Hardening

AWS Web Application Firewall (WAF) & AWS Shield

AWS Web Application Firewall (WAF): monitors/controls HTTP/HTTPS requests that are forwarded to CloudFront or an Application Load Balancer.

- Config includes: access based on IP, query string params.
- Offers 3 behaviours: (1) `ALLOW` (2) `BLOCK` (3) `COUNT`
- Additional protections based off: IP, Country, request header values, strings/regex in requests, request length, SQLi, XSS.

WAF deployment: done manually or via. CloudFormation template.

- Deploy WAF to CloudFront Distributions: global
- Deploy WAF to Application Load Balancer: region-specific

WebACL configuration example

- `CommonAttackProtectionManualIPBlockRule`: manually specify IPs to block
- `CommonAttackProtectionLargeBodyRule`: block requests w/ body size > limit
- `CommonAttackProtectionSqliRule`: block requests that indicate SQLi
- `CommonAttackProtectionXssRule`: block requests that indicate XSS

AWS Shield

- Basic-level turned on by default - $3,000/month for advanced-level.
- Advanced gives you an incident-response team + in-depth reporting.
- You won't pay if you are a victim of an attack.

EC2 Dedicated Instances vs. EC2 Dedicated Hosts

EC2 Dedicated Instances

- Run in a VPC on dedicated physical hardware separate from other AWS accounts, for a single customer.
- Dedicated instances may share hardware with other non-dedicated instances in the same AWS account.
- Billing: per-instance basis
  - On-demand.
  - Reserved Instances - save up to 70%.
  - Spot Instances - save up to 90%.

EC2 Dedicated Hosts

- Also runs on dedicated physical hardware from other AWS accounts, for a single customer.
- Provides additional visibility and control over how instances are placed on a physical server.
- Consistently deploy instances to the same physical server each time.
- Enable you to use your existing server-bound software licenses (e.g. VMWare, Oracle licenses which might require dedicated hosts).
- Enable you to address corporate and regulatory compliance.
- Billing: per-host billing

Provision Dedicated Instances / Dedicated Hosts via. EC2 service when launching an instance.

AWS Hypervisors, Isolation of AWS Resources, AWS Firewalls

AWS Hypervisor

- Hypervisor or virtual machine monitor (VMM) is software, firmware, hardware that creates an runs virtual machines.
  - Host machine: a computer on which a hypervisor runs 1+ virtual machines
  - Guest machine: each virtual machine
- EC2 runs on **Xen Hypervisors**: they can have guest OSs' running Paravirtualisation (PV) or using Hardware Virtual Machine (HVM).

- HVM guests are fully virtualized: VMs on top of hypervisors are not aware that they are sharing processing time with other VMs.
- PV is a lighter form of virtualisation and it used to be quicker.
- Performance gap between HVM/PV is closed and AWS recommends using HVM over PV.
- Windows EC2 instances can only be HVM where Linux can be HVM/PV.

- Paravirtualised guests
  - Relies on the hypervisor to provide support for operations that normally require privileged access.
  - Guest OS has no elevated access to the CPU.
  - CPU provides 4 separate privilege modes: 0-3 **"rings"**.
  - Host OS executes in **Ring 0**
  - Guest OS runs in lesser-privileged **Ring 1** and applications in least-privileged **Ring 3**
  - E.g. `R0: Xen Hypervisor` | `R1: Linux instance` | `R3: Applications`

What happens when we interact with EC2:

1. Physical Interface
2. Firewall splits traffic (runs at Hypervisor-layer - AWS managed)
3. Traffic is split and isolated through our security groups, our virtual interface, the hypervisor back to our resources.

Hypervisor Access (by AWS employees)

- Administrators with a business need to access the management plane requires MFA to access the administration hosts.
- The administration hosts are systems that are specifically designed, built, configured and hardened to protect the management plane of the cloud.
- All access is logged and audited.
- When an employee no longer has business need to access the management plane, privileges and access to these hosts can be revoked.

Guest OS (EC2) Access (by customers)

- These virtual instances are controlled completely by customers.
- Full root access over accounts, services and applications running on the EC2.
- AWS have no access rights to our Guest OS in EC2.

Memory Scrubbing:

- EBS automatically resets every block of storage used by the customer, so one customer's data is never unintentionally exposed to another customer. (all storage and RAM memory)
- Memory allocated to guests is scrubbed/zeroed by the Hypervisor when it is unallocated to a guest.
- Memory is not returned to the pool of free memory available for new allocations until scrubbing is complete.
- I.e. disk-recovery tools to find other customer's data won't work.

KMS Grants

KMS Grants are an alternate access control mechanism to a Key Policy

- Programmatically delegate use of KMS CMKs to other AWS principals (another user in your account / another account)
- Provide temp granular permissions (encrypt, decrypt, re-encrypt, describekey etc.)]
- Only grants ALLOWs, not DENYs
- Use Key Policies for static permissions, Grants for temp permissions.
- *Analogy: I give house keys to a friend to take care of my plants while I'm on holidays.*

KMS Grants are configure programmatically via CLI

- *create-grant*: adds new grant to CMK, specifies who can use it and list of operations the grantee can perform. A grant token is generated and can be passed as an argument to a KMS API.
- *list-grants*: lists grants
- *revoke-grant*: remove a grant

Example: Providing "Encrypt" operation as grant to IAM user

```
#Create a new key and make a note of the region you are working in
aws kms create-key

#Test encrypting plain text using my new key:
aws kms encrypt --plaintext "hello" --key-id <key_arn>

#Create a new user called Dave and generate access key / secret access key
aws iam create-user --user-name dave
aws iam create-access-key --user-name dave

#Run aws configure using Dave's credentials creating a CLI profile for him
aws configure --profile dave
aws kms encrypt --plaintext "hello" --key-id <key_arn> --profile dave
```

```
#Create a grant for user called Dave
aws iam get-user --user-name dave
aws kms create-grant --key-id <key_arn> --grantee-principal <Dave\'s_arn> --
operations "Encrypt"

#Encrypt plain text as user Dave:
aws kms encrypt --plaintext "hello" --key-id <key_arn> --grant-tokens
<grant_token_from_previous_command> --profile dave

#Revoke the grant:
aws kms list-grants --key-id <key_arn>
aws kms revoke-grant --key-id <key_arn> --grant-id <grant_id>

#Check that the revoke was successful:
aws kms encrypt --plaintext "hello" --key-id <key_arn> --profile dave

https://docs.aws.amazon.com/cli/latest/reference/kms/create-grant.html
```

KMS Policy Conditions - ViaService

Policy Conditions can be used to specify a condition within a Key Policy or IAM Policy

KMS provides a set of predefined **Condition Keys**.

- See https://docs.aws.amazon.com/kms/latest/developerguide/policy-conditions.html.

Use **kms:ViaService** to allow or deny access to your CMK according to which service the request originated from.

- Only for services that are integrated with KMS e.g. S3, EBS, RDS, Systems Manager, SQS, Lambda

ViaService example: CMK may be used for "Encrypt" action ONLY if request comes from EC2/RDS from the specified regions

```
"Effect": "Allow",
"Principal": {
    "AWS": "arn:xxx:xxx:xxx/ExampleUser"
},
"Action":[
    "kms:Encrypt",
]
"Resource":"*",
"Condition":{
    "StringEquals":{
        "kms:ViaService":[
            "ec2.us-west-2.amazonaws.com",
            "rds.us-west-2.amazonaws.com",
        ]
    }
}
```

KMS Cross Account Access for CMKs

2 steps to provide cross-account access.

- Example: Users in account HELLO need to use a CMK in account WORLD

1. Change the Key Policy for the CMK in account WORLD to allow ROOT USER in HELLO to have access. (doesn't have to be root account, can specify a specific user/role ARN instead)
2. Set up an IAM user/role in HELLO with explicit permission to use the CMK in WORD.

Example IAM policy in account HELLO for cross account access to CMK in WORLD

```
{
    "Statement":[
        {
            "Sid": "AllowUseOfCMKInAccountWORLD",
            "Effect": "Allow",
            "Action":[
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*"
            ],
            "Resource": "arn:aws:kms:us-west-2:WORLD:key/guid"
        }
    ]
}
```

Microservices

Monolithic applications

- Difficult to change.
- Can't make it bigger.
- One small mistake affects entire application.

Microservices

- Software is composed of small, independent services that communicate over well-defined APIs.
- Modern apps are usually made up of containers - a standardised unit which includes everything that your software needs to run e.g. libraries, system tools, runtime environment.

(ADVANTAGE #1) Serviceability: Easy to fix problems

- if one component breaks... Not A Disaster: the rest of the application keeps running.
- if one component breaks... Quick To Fix: deploy a new instance of your microservice/container to replace the broken one.

(ADVANTAGE #2) Flexibility: Easy to make changes

- to upgrade a Shopping Cart feature... you only need to replace the shopping cart microservice / deploy new containers.
- to add new features... e.g. add a product search feature, just add it as a new microservice. No need to redeploy the entire application.

(ADVANTAGE #2) Scalability: Easy to scale

- to scale the Shopping Cart microservice due to increased customer demand... just add more containers running Shopping Carts.
- Scale only components you need to = highly flexible and cost efficient applications.

Containers in AWS

Containers are a virtual operating environment.

- A standardised unit with everything that the software needs to run e.g. libraries, system code and runtime.
- Used to support microservices architecture.
- Use Docker to create Linux containers.
- Use Windows Containers for Windows workloads.

Architecture of a Docker container

- Container components:
  - (1) Code (2) Libraries (3) Virtual Kernel.
  - Runs on Docker.
  - Installed on host Operating System.
- Scaling application or building new features for the application = add more containers.

Where to run containers in AWS

- **Elastic Container Service (ECS)**
  - Fargate is the preferred option - Serverless.

  - Or managed clusters of EC2 instances.
  - Deep integration with AWS services e.g. IAM, VPC, Route53.
  - Used internally e.g. amazon.com, Sagemaker, Amazon Lex.
- **Elastic Kubernetes Service (EKS)**
  - Fargate is the preferred option - Serverless.
  - Or managed clusters of EC2 instances.
  - Certified Kubernetes conformant.
  - Benefit of open-source tooling from the community.
- Both above services are used for running and orchestrating containers and are a fully-managed Platform-As-A-Service (PaaS) service offering e.g. no need to install Docker, configure clusters, managing shared storing etc.
- Both provide a managed environment to run your containers.
- ECS has deep integration with AWS services vs. EKS benefits from the open-source community.

Elastic Container Service (ECS)

1. Container definition: choose a container image.
2. Task definition: describes your container attributes.
   - VPC, task execution role, Fargate/EC2, task memory, task CPU.
   - You can group multiple containers under a single task.
3. Service definition: a services allows you to run and maintain a specified number of simultaneous instances of a desk definition in an ECS cluster.
   - This makes sure an application remains up and running if something fails, as another instance of the task will be launched.
4. Configure cluster.

Container Security

1. Don't store secrets

- Use IAM roles instead of hardcoding user credentials.
- Use Secrets Manager for RDS credentials, API keys.
- Use Amazon Certificate Manager (ACM) if you have TLS certificates to store and manage.

2. Don't run as root

- Don't run containers using your AWS Root account.
- Don't run containers in EC2 as Root.

3. Less is more

- Minimise your attack surface: only run one service in your container.
- Avoid unnecessary libraries: remove code/libraries you don't need in your container image.

4. Use trusted images only

- Avoid public repositories, where you don't know the origin of the code.
- Use images from a trusted source or ones created in-house.
- Scan for CVE's using Amazon Inspector or external tools.
- **AWS Elastic Container Registry (ECR)**: a container registry where you can store your own container images e.g. Docker or Windows Container images and make them available to ECS.
    - AWS provides image scanning for container images stored in ECR and reports on any CVEs.

5. Infrastructure security

- Avoid the public internet use ECS Interface Endpoints (similar to VPC endpoints).
- If you must use public internet, Use TLS to secure end-to-end communication between end-users and your applications running in containers.
- If you are using TLS certificates, best-practice is to use **Amazon Certificate Manager (ACM)** as it provides a single, central interface for storing and managing certificates and it integrates well with many AWS services.
- Read https://aws.amazon.com/blogs/compute/maintaining-transport-layer-security-all-the-way-to-your-container-part-2-using-aws-certificate-manager-private-certificate-authority/.

Data Protection With VPCs

VPC Overview

Virtual Private Cloud (VPC) lets you provision a logically isolated section of the AWS cloud where you can launch AWS resources in a virtual network that you define / i.e. a LOGICAL DATACENTRE IN AWS.

- Have complete control over the virtual networking env.
    - Selection of your own IP range.

    - Creation of subnets.
    - Config of route tables and network gateways.
- Can easily customize network config for your VPC
    - e.g. create public-facing subnet for your webservers that has access to the internet.
    - e.g. create private-facing subnet for your databases/appservers that has NO access to the internet.
- Can leverage multiple layers of security
    - Security groups, network access control lists (NACLs) to help control access to EC2s in each subnet.
    - NOTE: SGs = stateful (port changes apply to inbound AND outbound) / NACLs = stateless (port changes apply to inbound OR outbound)
- Can create a Hardware Virtual Private Network (VPN) connection between your corporate datacenter and VPC and leverage AWS cloud as an extension of your corporate datacenter.
- 1 SUBNET = 1 AVAILABILITY ZONE
- MINIMUM IPs per subnet: subnet /28 = 16 ip addresses
- MAXIMUM IPs per subnet: subnet /16 = 65,536 ip addresses
- You can have multiple VPCs inside a region.

How VPC works:

1. Traffic entry into VPC:
    - via. **VPC Virtual Private Gateway** (VPN connection)
    - via. **VPC Internet Gateway** (internet connection)
2. Traffic to Router:
    - Configure how this traffic is routed via. Route Tables
3. Traffic hits Network ACLs (first-line of defence)
4. Traffic hits Security Groups
    - SGs govern if traffic is allowed to talk to our instances.
5. Traffic hits destination EC2

What can you do with a VPC?

- Launch instances into a subnet of your choosing.
- Assign custom IP address ranges in each subnet.
- Configure route tables between subnets.
- Create internet gateway and attach it to your VPC.
- Much better security control over your AWS resources.

Default VPC vs. Custom VPC:

- Default VPC is user friendly, allowing you to immediately deploy instances.
- All subnets in default VPC have a route out to the internet.
- Each EC2 instance has both a public and private IP address (unless they're in private subnets).

VPC Peering: connect one VPC with another via. direct network route using private IP addresses.

- Instances behave as if they were on the same private network.
- You can peer VPCs with other AWS accounts as well as other VPCs in the same account.
- Peering is in a star configuration: i.e. 1 central VPC peers with 4 others.
    - NO TRANSITIVE PEERING e.g. VPC A can talk to B vice-versa, but A cannot talk to C via. B. Peering must be created between A and C.

Setting up a Custom VPC

1. (NOT REQUIRED) Check out the default resources in your AWS account
    - Default VPC, default subnets, default route table, default internet-gateway, default security group.
2. Go to VPC -> Create a new VPC
    - IPv4 CIDR block: 10.0.0.0/16 (biggest address range possible)
    - Tenancy: default (multi-tenant hardware environment) / dedicated (no sharing with other AWS customers, expensive.)
    - You should now have the resources provisioned: a Route Table, Network ACL, Security Group (won't create default subnets)
3. Provision a series of subnets
    - Name: [CIDR range]-[availability zone]
    - Select the VPC you created, select the AZ, CIDR address range
4. Provision an Internet Gateway - so there is connectivity from the internet to the VPC
    - Create an IG -> attach to the custom VPC.
    - Each VPC can only have ONE IG attached.
5. Create a new Route Table that is EXPLICITLY public-facing into the Custom VPC - to replace the default internet-facing MAIN Route Table.
    - Enable internet-access: Goto `Routes` -> add route `0.0.0.0/0` -> select target Internet Gateway.
    - Associate subnet with the new Route Table: Goto `Subnet Associations` -> select a subnet to associate the new Route Table with.

- Disable internet-access for MAIN Route Table, so all new subnets created by default won't be internet-facing anymore.

6. Test internet connectivity - using EC2s
    - First goto `Subnets` -> `Subnet actions` -> turn on `Auto-assign public IP addresses` for the public subnet
    - TEST PUBLIC (representing a webserver): Launch an EC2 -> select Custom VPC -> select the public subnet -> create SG with open port 22.
    - TEST PRIVATE (representing a private server e.g. SQL server): Launch an EC2 -> select Custom VPC -> select the private subnet.
    - You should be able to SSH into the public EC2 using the public IP address.
    - You should NOT be able to SSH into the private EC2 as there is no assigned public IP / SG config does not allow.
7. Configure private EC2 server - example is a MYSQL server
    - Allow inbound 22 SSH, 3306 MYSQL, 80/443 HTTP(S), 0-65535 ICMP PING with source = public subnet CIDR address (so EC2 in private subnet can talk to the webserver in the public subnet)
8. Connect to the private EC2 from public EC2
    - Test connection by SSH into public EC2 and PING private EC2.
    - *(SECURITY WARNING: IN PRODUCTION, USE A BASTION HOST)* Store private key for private EC2 on the public EC2 -> SSH into private EC2.
    - NOTE: there will be no outbound route yet to the public from the private EC2.
9. Create outbound route for the private EC2 w/o placing into public subnet - for installing packages / patch OS.

The first 4 IP addresses and the last IP address of each subnet CIDR block can't be used as:

- IP #1 reserved for the network address
- IP #2 reserved for the VPC router
- IP #3 reserved for DNS purposes `10.0.0.2`
- IP #4 reserved for future use
- IP #LAST reserved for network broadcast address

SUBNETS: *The purpose of subnetting is to help relieve network congestion. If you have an excessive amount of traffic flow across your network, then that traffic can cause your network to run slowly. When you subnet your network, most of the network traffic will be isolated to the subnet in which it originated. Ideally, your subnet structure* *should mimic your network's geographic structure. Other benefits of subnetting include routing efficiency, easier network management and improving network security*

ROUTE TABLES: *Route tables contain a set of rules (routes) that are used to determine where network traffic from your subnet or gateway is directed. It allows subnets to talk to each other. Every AWS subnet you provision will automatically be attached to your default/main route table. Ideally, you should create a separate route table that is internet accessible rather than use your default/main route table as every new subnet being provisioned will be associated with the default/main route table, hence become internet accessible.*

NAT Instances (OLD METHOD)

NOTE: Network Address Translation (NAT) is a process where a network device assigns a public IP address to a computer inside a private network. The purpose of a NAT is to limit the no. of public IP addresses a company must use for economic and security purposes.

Launch and set up a NAT ec2 instance

1. Find a NAT instance within community AMI's
2. Place instance in the custom VPC.
3. Place instance in the public subnet.
4. Configure the Web-DMZ security group (with SSH22/HTTP80/HTTPS443 open).
5. Launch the instance.
6. Configure instance to *disable Source/Destination checks* (used by normal ec2s) as a NAT instance is not the src/dest itself.

Create a route OUT from the default route table via. NAT instance:

1. Goto VPC -> Route Tables -> select default route table
2. Edit default route table -> add destination `0.0.0.0/0`, with the `NAT instance` as the target.

Test the route out:

1. SSH into public instance -> SSH into private instance using key.
2. Ping google or run `yum update` to test internet accessibility.

NAT instance downsides:

- Bottlenecks: single instance, single availability zone, limited network throughout.
- The amount of traffic that NAT instances can support depends on instance size. You must increase instance size if more throughout is required.
- Reliant on a single OS, any crashes = no internet access for any servers in the private subnet.
- High availability requires using Autoscaling Groups, multiple subnets in different AZs and scripts to automate failover = pain in the ass.
- Bad design in general to use NAT instances, as it can get complex to make it work efficiently.
- **AWS new feature NAT gateway now should replace the use of a single NAT instances.**

NAT instance can be used as a bastion server (server used to RDP/SSH into instances within your private subnet).

NAT Gateways (PREFERRED METHOD)

Launch and set up a NAT gateway

1. Goto VPC -> select NAT Gateway
2. Select the `public subnet` in the custom VPC -> Select `create new EIP` to create an Elastic IP address.
3. Click `create a NAT Gateway`
4. Edit default route table -> add destination `0.0.0.0/0`, with the `NAT gateway` as the target.

Test the route out by performing the same test as with the NAT instance.

Comparison of NAT instances and NAT gateways: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-comparison.html

Benefits of NAT Gateways:

- Preferred by enterprise.
- Scales automatically up to 10Gbps.
- Highly available, automatic failover.
- NAT Gateways are managed by AWS (patching, antivirus etc. = more secure than NAT instances)
- NAT Gateways don't need to sit behind a security group.
- Automatically assigned with a public IP (no need to create EIP).

Having 1 NAT Gateway in 1 AZ is not good enough, you need to have at least 1 NG per AZ so there is some form of redundancy in terms of AZ failure.

NACLs vs. Security Groups

Network Access Control List (NACL) acts as a firewall for controlling traffic in/out of your subnets.

- NACLs are stateless, responses to allowed inbound traffic are subject to the rules of outbound traffic (vice versa.)
- You can only associate 1 subnet to 1 NACL, not 1 subnet to multiple NACLs
- Subnets are automatically associated with the default VPC NACL.
- NACLs can only be deployed to 1 VPC, they cannot span VPCs.
- The *default VPC NACL* will *ALLOW ALL* traffic in/out of subnets associated with the NACL.
- Any *custom NACLs* created by default will *DENY ALL* traffic in/out.

Creating and configuring NACL (example: setting up webserver):

1. Goto VPC -> Network ACLs -> `Create Network ACL`
2. Add inbound rules `HTTP 80`, `HTTPS 443`, `SSH 22` ALLOW, leaving the DENY ALL.
3. Add outbound rules `HTTP 80`, `HTTPS 443`, `Custom TCP 1024 - 65535` ALLOW, leaving the DENY ALL.
4. Associate NACL with the public subnet. Since 1 subnet can only be associated with 1 NACL, the default NACL will be disassociated.

Rules are evaluated in numerical order.

- Example: Rule `#100 | HTTP 80 ALLOW ALL` vs. Rule `#101 | HTTP 80 DENY MY_IP`
  - The website will work because Rule #100 overrides Rule #101.
- Example: Rule `#100 | HTTP 80 ALLOW ALL` vs. Rule `#99| HTTP 80 DENY MY_IP`
  - The website WON'T work anymore because Rule #99 overrides Rule #101.

NACLs are assessed BEFORE Security Groups - traffic blocked on NACL level won't reach SG, even if SG allows HTTP80.

You can block IP addresses using NACLs, not Security Groups.

Application Load Balancers and Custom VPC's

Setting up an ALB (a type of Elastic Load Balancer - AWS offers ELB options Application/Network/Classic)

- Goto EC2 -> `Create Load Balancer` -> `Application Load Balancer`
- Select Scheme = `internet facing`
- Select subnets from at least TWO public Availability Zones to increase the availability of your Load Balancer.

Elastic Load Balancers and TLS/SSL Termination

When using ELBs, you have the choice to terminate TLS/SSL on the Load Balancer or EC2 instances.

Terminate at load balancer

- ALB decrypts HTTPS request -> inspects HTTP headers -> routes request to EC2 as plaintext over the local private network in your VPC.
- Benefits
  - Offloads decryption overhead to ALB, meanings EC2 has more resources for application processing.
  - More cost effective as you require less EC2 compute power, therefore can use smaller EC2 instances to handle application load.
  - Reduces administrative overhead if you have many EC2 instances, from managing X509 certificates (used to encrypt/decrypt) individually on multiple EC2s.
- Security implications
  - Traffic between ALB and EC2 is unencrypted (however, AWS states that network traffic cannot be listened to by EC2s that aren't part of the connection, even if they are running within your own AWS account).
  - Compliance / regulatory requirements to use end-to-end encrytion all the way to your EC2 may require you to terminate TLS/SSL on the EC2 instances.

Which Load Balancer to use?

- Application Load Balancer only supports TLS/SSL termination on the Load Balancer itself. Only supports HTTP/HTTPS.
- Network Load Balancer supports TLS/SSL termination on your EC2 instances. You will need to use TCP protocol (load balancing at the TCP level).
- Classic Load Balancer is a legacy option.

Exam tips:

- Best use of EC2 resources = use APPLICATION.
- Regulatory / Compliance requirements for E2E-encryption = use NETWORK or CLASSIC.
- For any other protocol that is not HTTP/HTTPS = use NETWORK or CLASSIC.

VPC Flow Logs

VPC Flow Logs enables you to capture info about the IP traffic going to/from network interfaces (ENIs) in your VPC.

- Flow Log data is stored using AWS CloudWatch logs.
- Flow Logs can be created at 3 different levels:
  i. VPC level: capture all ENI traffic
  ii. Subnet level: capture ENI and EC2 traffic within a particular subnet
  iii. Network Interface level

Creating Flow Logs

1. Goto VPC -> Click `Actions` -> `Create Flow Log`.
2. Select `Filter` (type of traffic to log) = ALL traffic, ACCEPTED traffic , REJECTED traffic.
3. Select `Role` = Flow Log IAM role that allows Flow Log to create logs in CloudWatch.
4. Select `Destination Log Group` = goto CloudWatch -> create a Log Group -> select this group.

Flow Log options

- You can stream the logs to AWS Lambda or AWS Elasticsearch.
  - You can have your environment pro-actively react to something that happens inside your VPC.
- You can export the data to S3.

Exam tips;

- You cannot enable Flow Logs for VPCs that are PEERED with your VPC unless the peer VPC is in your account.
- You cannot tag a Flow Log.
- After creating a Flow Log, you cannot change its configuration e.g. can't associate to another IAM role with the Flow Log.
- Not all IP traffic is monitored:

- Traffic generated by instances when they contact the AWS DNS server. Logged only if you use your own DNS server.
- Traffic generated by Windows instance for Amazon Windows license activation.
- Traffic to/from `169.254.169.254` for instance metadata.
- DHCP traffic.
- Traffic to reserved IP addresses (1st four and last IP) for the default VPC router.

NATs and Bastions

NAT instance: used to provide internet traffic to EC2 instances in private subnets. Bastion instance (jump boxes): used to securely administer EC2 instances (using SSH/RDP) in private subnets.

How to build a highly available Bastion instance:

- High availability: at least 2x Bastion Instances in 2 public subnets in 2 AZ.
- Autoscaling Groups: minimum of 1 Bastion, if Bastions goes down, ASG will deploy a Bastion into one AZ or the other.
- Route53 running health checks on the Bastion server.

Highly available NAT instances will have a similar approach as Bastion instances above.

NAT Gateways will automatically handle failover.

Session Manager in AWS Systems Manager

Session Manager enables secure remote login to EC2 instances - alternative to SSH/RDP but more secure.

Simple: manage both Windows/Linux instances with the same tool Remote Login: browser-based, run an interactive session using Powershell/Bash. Secure:

- TLS encryption;
- No Bastion hosts.
- No opening inbound ports required. Everything is logged
- Connection history recorded in CloudTrail.
- Keystroke logging and sent to CloudWatch/S3.

Setting up Session Manager in AWS Systems Manager service

1. Create an IAM role to enable EC2 to call Systems Manager (Policy: `AmazonEC2RoleforSSM`)
2. Launch an EC2 using the role, with a Security Group that has no rules (since we don't need to open ports)
3. Create a CloudWatch Log Group for Session Manager `SM_LogGroup`.
4. Configure Session Manager #1: Goto `Systems Manager` -> `Session Manager` -> `Preferences` -> Enter the CloudWatch Log Group name you created above.
5. Configure Session Manager #2: Choose from logging options

- Encrypt session logs with KMS
- Send session logs to an S3 bucket
- Send session logs to CloudWatch logs.

6. Start a sessions: Goto `Sessions` -> `Start a session` -> Select running EC2 instance to launch web shell.

SSM-user has root privileges by default. You can view session history / all the commands that were run during the session including all the output.

VPC Endpoints

VPC Endpoint enables you to privately connect VPC to supported AWS Services, without needing to go through a NAT Gateway - it goes over the private network, instead of the public network.

- Normal: VPC internal network -> NAT Gateway -> AWS S3.
- VPC Endpoint: VPC internal network -> internal gateway -> AWS S3.

Creating a VPC Endpoint

1. Create an IAM role to enable EC2 to call S3 (Policy: `AmazonS3FullAccess`)
2. Goto `EC2` -> change an attached EC2 role to the new role created above
3. Goto `VPC` -> `Endpoints` -> `Create Endpoint` -> select the S3 service gateway -> select the VPC you want to have the gateway -> select the appropriate Route Table associated with the private subnet.

You can now see the VPC Endpoint route in the chosen Route Table. You can also SSH into private EC2 and run `aws s3 ls` to test the route.

AWS CloudHSM

CloudHSM

- Tenancy: single-tenancy i.e. physical device is dedicated to you.
- Key control: you control all the keys i.e. only person who can access keys is yourself, not AWS.
- Symmetry: symmetric and asymmetric keys are available.
- Compliance: FIPS 140-2 | EAL-4 compliant.
- More expensive: $ charged by the hour.

AWS CloudHSM provides hardware security modules (HSMs) in a cluster. A cluster is a collection of individual HSMs that AWS CloudHSM keeps in sync. You can think of a cluster as one logical HSM. When you perform a task or operation on one HSM in a cluster, the other HSMs in that cluster are automatically kept up to date.

Creating and setting up a HSM Cluster:

1. Create a VPC + public and private subnet (so Cluster is across multiple Availability Zones for HA)
2. Create the Cluster

- Creation of Cluster will create a new Security Group with open inbound ports that CloudHSM will use to communicate with our EC2 instances.

3. Verify HSM Identity (optional)
4. Initialise the Cluster
5. Launch a client EC2 instance
6. Install and configure the client software on the instance
7. Activate the Cluster
8. Setup Users + Generate Symmetric/Asymmetric Keys

CloudHSM User Types:

1. *Precrypto Officer (PRECO)*: default account with admin/password creds -> upon password setting, you will be promoted to CO.
2. *Crypto Officer (CO)*: performs user management operations e.g. create and delete users and change user passwords.
3. *Crypto Users (CU)*: performs key and crypto management operations

- Key managent - create, delete, share, import, export cryptographic keys.
- Cryptographic operations - use cryptographic keys for encryption, decryption, signing, verifying and more.

4. *Appliance User (AU)*: perform cloning and synchronization operations.

- CloudHSM uses the AU to synchronize HSMs in the AWS CloudHSM Cluster.
- AU exists on all HSMs provided by AWS CloudHSM and has limited permissions.s

Check out https://docs.aws.amazon.com/cloudhsm/latest/userguide/hsm-users.html for more info of CloudHSM users.

```
# Before you can initialize a Cluster, you must download and sign a Certificate
Signing Request (CSR) that is generated by the Cluster's # first HSM.

# generate private key "customerCA.key"
openssl genrsa -aes256 -out customerCA.key 2048
# generate public key (using the private key) - this is the self-signed
certificate
openssl req -new -x509 -days 3652 -key customerCA.key -out customerCA.crt
# copy the cert signing request "ClusterCsr.csr" downloaded from AWS CloudHSM into
your instance
nano <cluster_id>_ClusterCsr.csr
# create signed Cluster certificate, used to initialize our Cluster: take
private/public keys to sign the ClusterCsr.csr
openssl x509 -req -days 3652 -in <cluster_id>_ClusterCsr.csr \
                              -CA customerCA.crt \
                              -CAkey customerCA.key \
                              -CAcreateserial \
                              -out <cluster_id>_CustomerHsmCertificate.crt

# get the software client to help administer the HSM
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-
client-latest.el7.x86_64.rpm
# install the client
sudo yum install -y ./cloudhsm-client-latest.el7.x86_64.rpm
# copy the public key to the local CloudHSM directory
cp customerCA.crt /opt/cloudhsm/etc/customerCA.crt

# configure the Cluster with private IP of the Cluster
sudo /opt/cloudhsm/bin/configure -a <cluster_IP>
# CONNECT TO + MANAGE THE CLUSTER using the tool "cloudhsm_mgmt_util"
/opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg

# enable end-to-end encryption
enable_e2e
# ret list of users configured on CloudHSM
listUsers

# setup PRECO and setup password
loginHSM PRECO admin password
changePswd PRECO admin <NewPassword>
listUsers
logoutHSM

# login as CO -> create CU
loginHSM CO admin acloudguru
createUser CU ryan acloudguru
listUsers
logoutHSM
quit
```

```
# start managing keys by starting "cloudhsm-client" service -> "key_mgmt_util"
tool
sudo service cloudhsm-client start
/opt/cloudhsm/bin/key_mgmt_util
# login as CU -> generate symmetric key + asymmetric keypair
loginHSM -u CU -s ryan -p acloudguru
genSymKey -t 31 -s 32 -l aes256
genRSAKeyPair -m 2048 -e 65537 -l rsa2048
# generate a WRAPPING KEY to prepare for exporting symmetric and/or priv keys
genSymKey -t 31 -s 16 -sess -l export-wrapping-key
# export Symmetric / Asym privkey using the wrapping key
exSymKey -k <symmetric_key> -out aes256.key.exp -w <wrapping_key>
exportPrivateKey -k <private_key> -out rsa2048.key.exp -w <wrapping_key>
# export the Asym pubkey (no need for wrapping key)
exportPubKey -k 22 -out rsa2048.pub.exp
logoutHSM
exit
```

Exam Tip: Remember the 4 user types: PRECO | CO | CU | AU

AWS DNS and Custom VPCs

Creating a VPC = automatically includes an AWS DNS server which is used to public DNS hostnames.

- Used for instances in your VPC which are communicating over the internet.
- DNS server uses (one of the five) reserved IP address in your VPC CIDR range - `10.0.0.2`

Using your own custom DNS server.

1. Disable the AWS DNS server: Select `your VPC -> actions -> edit DNS Resolution -> Uncheck checkbox.`
2. Use your own custom DNS: Goto `DHCP options set -> Create DHCP options set` -> fill in fields -> associate with your VPC.

AWS Transit Gateway

VPC connectivity can be very messy. AWS Transit Gateway service helps simplify your network when you have multiple VPCs and your own datacentre and you need everything to communicate with each other (via. VPC peering).

Non-Transit Gateway

- Each VPC requires VPN connection and configuration to the On-Prem Network / Datacentre.
- VPCs require peering between each other.
- Hundreds of VPCs: difficult to manage, not scalable.

Transit Gateway

- Highly scalable: supports thousands of VPCs (hub-and-spoke architecture)
- Centralised: Transit Gateway sits between all your VPCs and Datacentre. Only need to configure once. Any VPC connected via. Transit Gateway can communicate with every other connected VPC.
- Route Tables are used to control which VPCs can communicate with each other.
- Secure: communication between VPCs are done via. AWS private network. Inter-region traffic is supported.

AWS VPC Summary

MAKE SURE YOU CAN BUILD A CUSTOM VPC FROM MEMORY BEFORE TAKING EXAM.

Summary:

- Build a custom VPC with **private + public subnet**.
- Instances in private subnet have internet access via. **NAT Gateway**.
- Replace **DEFAULT Route Table** (which all new subnets are associated with) with a **NEW Route Table** and put a route out using **Internet Gateway** -> Every subnet we want to make public, we would associate with that NEW Route Table.
- Create a **NAT Instance** -> disable Src/Dest check.
  - They must be in a public subnet.
  - There must be a route out of private subnet to the NAT instance for this to work.
  - Amount of traffic NAT instance supports depends on instance size. Bottleneck = increase EC2 size.
  - You can create highly availability NAT instances using ASGs, multiple subnets in different AZs, script to automate failover.
- ^**NAT Gateway** is better than NAT Instance
  - Scale automatically up to 10Gbps
  - No patching, no associated Security Groups, automatically assigned public IP address.
  - Remember to update Route Tables when provisioning NAT Gateways.
- **Network Access Control Lists (NACL)**
  - Create a VPC: default NACL allows all inbound/outbound by default.
  - Custom NACL: denies all inbound/outbound by default until you add rules.

  - Each subnet in your VPC MUST be associated with a NACL. If not, it will be automatically associated with your default Network ACL.
  - You can associate NACL with multiple subnets, however each subnet can have only one NACL.
  - NACLs contain a **numbered list of rules evaluated in order**, **starting with the LOWEST numbered rule first**.
  - NACLs are stateless: responses to inbound traffic are subject to the rules of outbound traffic vice versa.
  - NACLs can **block IP addresses**, Security Groups cannot.
- **Application Load Balancer**
  - You need at least 2 public subnets in order to deploy an ALB.
- **VPC Flow Logs**: monitoring network traffic across ENIs
  - You cannot enable flow logs in VPCs that are peered with your VPC unless peering is within your account.
  - You cannot tag a flow log.
  - You can't change flow log config after creating a flow log e.g. can't associate with different IAM role.
  - Not all IP traffic is monitored:
    - Traffic generated by instances when contacting AWS DNS server.
    - Traffic generated by Windows instance for AWS Windows license activation.
    - Traffic to/from instance metadata calls `169.254.169.254`.
    - DHCP traffic.
    - Traffic for reserved IP addresses for default VPC router.
- **VPC Endpoints**: Bypass NAT Gateway (public network) to access AWS service directly.

Incident Response & AWS In The Real World

Distributed Denial of Service (DDoS) Overview

Recommended to read DDoS Whitepaper before exam, to prepare for DDoS QNs: https://d0.awsstatic.com/whitepapers/Security/DDoS_White_Paper.pdf

DDoS is worth quite a few points in the exam

DDos is an attack that attempts to make your website or app unavailable to your end users via. multiple methods such as:

- Large packet floods.
- Combination of reflection and amplification techniques.
- Using large botnets.

Amplification / Reflection attacks include NTP/SSDP/DNS/Chargen/SNMP attacks etc.

- Attacker sends a 3rd-party server a request using a spoofed IP -> server responds with a greater payload than initial request.
- Response payload is usually 28-54 times larger to the spoofed IP address.
- Example 1: NTP Amplification
    - Attacker sends 64 byte request with spoofed IP -> server responds by sending 3,456 bytes of traffic to spoofed IP.
    - Attacker co-ordinates this with multiple NTP servers a second to send legit NTP traffic to the target.
- Example 2: **Slowloris** (application attack)
    - Attack opens multiple connections with server without closing them, by sending partial HTTP requests.
    - Server waits for connections to be completed -> server's max concurrent connections pool is filled -> drops legit traffic.

How to mitigate DDoS:

1. Minimize the Attack Surface Area.
    - Some prod environments have multiple entry points e.g. SSH/RDP to web servers, DB servers etc.
    - Use a **jump box / bastion host** + whitelist allowed IP addresses + move these servers to a private subnet.
2. Be ready to scale to absorb the attack.
    - Key strategy behind DDoS: bring your infra to breaking point. Defeat this strategy by designing your infra to scale as/when it is needed.
    - Scale **horizontally** (add more machines into pool of resources) and **vertically** (add more power to existing machines).
    - Attack is *spread over a large area*.
    - Attackers have to *counter attack*, taking up more of their resources.
    - Scaling buys time to *analyze* the attack.
    - Scaling provides you with *additional levels of redundancy*.
3. Safeguard exposed resources.

  - For situations where you can't eliminate internet entry points to your apps, take additional measures to restrict access without interrupting legitimate user traffic. Three measures:
  - AWS WAF: Most DDoS attacks are app-layer rather than infra-layer
    - **WAF Service**: protect EC2s, ALBs, CF distributions.
    - **AWS Marketplace WAFs**: use external 3rd-party WAFs.
  - AWS CloudFront:
    - **Geo Restriction/Blocking**: restrict access to users in specific countries (whitelist or blacklists).
    - **Origin access identity**: restrict access to your S3 bucket so that people can only access S3 using CloudFront URLs.
  - AWS Route53.
    - **Alias Record Sets**: Immediately redirect traffic to an AWS CF distribution, or to a different ELB with higher capacity EC2 instances running WAFs or your own security tools.
    - **Private DNS**: Allows you to manage internal DNS names for your app resources (web servers, databases) without exposing this info to the public internet.

4. Learn what normal behaviour looks like.
   - Be aware of normal and unusual behaviour.
   - Spot abnormalities fast -> create alarms to alert you of unusual behaviour -> collect forensic data to understand attacks.
5. Create a plan for attacks.
   - You've validated the design of your architecture.
   - You understand costs for increased resiliency and know what techniques to employ when an attack happens.
   - You know who to contact when an attack happens.
6. **AWS Shield**: service that protects all AWS customers on ELB, CloudFront and Route53.
   - Protects against SYN/UDP floods, reflection attacks and other layer 3/4 (Network and Transport) attacks.
7. **AWS Shield Advanced**: provides enhanced protections for your apps running on ELB, CloudFront, Route53 against larger and more sophisticated attacks. Costs $3,000 per month.
   - Always-on, flow-based monitored of network traffic and active application monitoring to provide near real-time notifications of DDoS attacks.
   - DDoS Response Team (DRT) 24/7 to manage and mitigate application-layer DDoS attacks.

- Protects AWS bill against higher fees due to ELB, CloudFront and Route53 usage spikes during DDoS attacks.

WAF Integration into AWS

WAF scenarios will be in the exam.

WAF only integrates directly with **(1) Application Load Balancers** , **(2) Amazon API Gateway** and **(3) CloudFront Distributions**. WAF does NOT integrate with EC2, DynamoDB, Route53 or any other services.

EC2 has been hacked - what to do?

1. Stop the instance immediately.
2. Take a snapshot of the EBS volume + terminate the instance.
3. Deploy a copy of the instance in a totally **isolated environment**.

- Isolated VPC, no internet access - ideally a private subnet.

4. Access the instance using an **isolated forensic workstation**.

- Don't do it on your normal laptop - use a dedicated workstation/device with an antivirus, no software on it except for forensic tools such as Wireshark, Kali etc.

5. Read logs to figure out how they obtained access.

Leaked Github keys - what to do?

For IAM Users:

1. Goto `IAM` -> De-activate the IAM User Access Key.
2. Create a new IAM User Access Key.
3. Delete the old IAM User Access Key.

For Root User:

1. Goto `My Security Credentials` (top nav / outside of IAM).
2. Goto `Access Keys` -> De-activate Root User Access Key.
3. Create a new Root User Access Key.
4. Delete the old Root User Access Key.

Reading CloudTrail Logs

1. Goto `CloudTrail` -> `View Trail`
2. Select a trail -> select `S3 log bucket` -> Choose region -> date -> open a log S3 object.

Exam tips:

- Understand that any API calls made in AWS are logged in CloudTrail.
- Replicate CloudTrail logs to an audit account which no-one else has access to.
- Any "performance" monitoring related questions would be CloudWatch, NOT CloudTrail.

Penetration Testing in AWS

Penetration Testing is allowed without prior approving for 8 services

1. EC2
2. RDS
3. CloudFront
4. Aurora
5. API Gateway
6. Lambda and Lambda Edge functions
7. Lightsail resources
8. Elastic Beanstalk environments

Prohibited Activities

1. DNS Zone walking via. Route53w Hosted Zones
2. DDoS
3. Port flooding
4. Protocol flooding
5. Request flooding (login request flooding, API request flooding)

Other Simulated Events

- Request authorization for other simulated events by emailing `aws-security-simulated-event@amazon.com`

AWS Certificate Manager (ACM)

Use **AWS Certificate Manager (ACM)** to provision a SSL certificate for a domain name you have registered. SSL certificates are automatically renewed provided you purchase the domain name from **Route53**.

You can import your own certificate vs. you can request a certificate.

Requesting a certificate

1. Add your domain name
2. Select your domain validation methods
    - (1) DNS validation: requires you to modify DNS config for the domain in your certificate request.
    - (2) Email validation: requires you to respond to an email sent to an email address under the domain.
3. For DNS validation
    - Add a CNAME record to the DNS config for your domain.
    - Goto `Route53` -> create a record -> add the CNAME record.
4. Wait ~5-10 minutes and the status of the certificate should be `Issued`.

SSL/TLS certificate renewal

- Auto-renewal: ACM provides autorenewal for Amazon-issued SSL/TLS certs.
- Manual renewal: Imported SSL/TLS certs OR certs associated with R53 private hosted zones must be manually renewed.

Using Amazon SSL certificates

1. SSL/TLS cert with CloudFront

- Goto `CloudFront` -> select a distribution -> select `Distribution Settings` -> edit to change from default CloudFront SSL cert to the new custom SSL certificate associated with your domain name.

2. SSL/TLS cert with EC2

- Goto `EC2` -> `Load Balancers` -> create a Load Balancer -> `choose a certificate from ACM`

NOTE: You CANNOT export Amazon-issued SSL/TLS certs and use it elsewhere, only within AWS services.

Securing Load Balancers using Perfect Forward Secrecy

Perfect Forward Secrecy A property of secure communications protocols in which compromises of long-term (public/private key) keys DO NOT compromise past session keys. Forward secrecy protects past sessions against future compromises of secret keys or passwords.

Security Policy when setting up ALB:

- Choose the `2016-08` Security Policy as it supports most ciphers.
- Enable Perfect Forward Secrecy on your ALBs by selecting a Security Policy with a `ECDHE-X` cipher.

API Gateway - Throttling and Caching

AWS API Gateway throttling

- API Gateway throttles requests to your API, to prevent it from being overwhelmed by too many requests.
- When request submissions exceed steady-state request rate or burst-limits, API Gateway fails the limit-exceeding requests and returns `429 Too Many Requests` to the client.
- Limits
    - `Steady-state` = 10,000 requests/second.
    - `Burst-limit` (max concurrent requests that API Gateway can fulfil) = 5,000 requests across all APIs within an AWS account.
- Examples
    - Caller submits 10,000 requests/second period evenly (e.g. 10 requests/ms) = ALL REQUESTS SERVED.
    - Caller submits 10,000 requests in the 1st millisecond = FIRST 5,000 SERVED -> THROTTLES REMAINING 5,000 FOR REMAINING 1 SECOND PERIOD.
    - Caller submits 5,000 requests in the 1st millisecond, then evenly spreads another 5,000 requests through remaining 999 milliseconds. = SERVES ALL REQUESTS IN 1 SECOND PERIOD WITHOUT 429 RESPONSE.
- Account-level rate limit and burst limit can be increased upon request e.g. Ticketmaster who will have huge traffic spikes.

AWS API Gateway caching

- Use API Caching in AWS API Gateway to cache endpoint's response.
- Use caching to reduce number of calls made to your endpoint and also improve latency of requests to your API.

- When caching is enabled for a stage (test, prod etc.), API caches responses from your endpoint for a specified **Time To Live (TTL)**.
- `TTL 300` = default | `TTL 3600` = maximum | `TTL 0` = caching disabled.

AWS Systems Manager - Parameter Store

AWS Systems Manager is a service to manage EC2 systems at scale.

1. Create a parameter

- Type: *String* (plaintext), *String List* (plaintext list), *Secure String* (KMS encrypted)

2. Store sensitive data inside
3. Access parameters across different AWS services.

- Accessed by EC2, EC2 Run Command, Lambda, CloudFormation etc.

Exam only requires high-level knowledge of AWS Systems Manager (Parameter Store)

AWS Systems Manager - EC2 Run Command

The Systems Manager (SSM) **EC2 Run Command** allows you to:

- manage a large number of EC2 instances and on-premise systems.
- automate admin tasks and adhoc config changes e.g installing apps, patching, joining new instances to a Windows domain without having to RDP into each instance.

Using EC2 Run Command:

1. Create a role for SSM - `EC2 role for Simple Systems Manager`
2. Create an instance - Windows image, attach IAM role created above.
3. Under `Actions` in SSM, click `Run a Command` -> choose a command document i.e. `Configure CloudWatch` -> select the target instance and then run.

Exam tips:

- Commands can be applied to a group of systems based on AWS instance tags or by selecting manually.
- **SSM agent** needs to be installed (it is installed by default on certain Windows and Linux AMIs) and an IAM SSM role enabled on all your managed instances for Run Command to work.

- The commands and parameters are defined in a **Systems Manager Document**
- Commands can be issued using AWS Console, AWS CLI, AWS Tools for Windows PowerShell, Systems Manager API or Amazon SDKs.
- You can use this service with your on-premise systems as well as EC2 instances.

Compliance Frameworks

ISO27001

- Specifies requirements for *establishing*, *implementing*, *operating*, *monitoring*, *reviewing*, *maintaining* and *improving* documented Information Security Management System (ISMS) within the context of the organization's overall business risks.

FedRAMP (Federal Risk and Authorization Management Program)

- Government-wide program that provides a standardised approach to security assessment, authorisation and continuous monitoring for cloud products and services.

HIPAA (Federal Health Insurance Portability and Accountability Act of 1996)

- Primary goal is to make it easier for people to keep health insurance, protect the confidentiality and security of healthcare info and to help the healthcare industry control administrative costs.
- Primary goal: *lower cost of healthcare* and *ensure good data security around people's healthcare info*.

NIST (National Institute of Standards and Technology - U.S Department of Commerce)

- A framework for improving critical infrastructure cybersecurity - a set of industry standards and best practices to help organisations manage cybersecurity risks.

PCI DSS (Payment Card Industry Data Security Standard)

- Widely accepted set of policies and procedures intended to optimise security of credit, debit and cash card transactions and protect cardholders against misuse of their personal info.

PCI DSS 12 requirements (not required for exam, but good for interviews):

- *Build and Maintain a Secure Network and Systems*

1. Install and maintain a firewall configuration to protect cardholder data.
2. Do not use vendor-supplied defaults for system passwords and other security parameters.

- *Protect Cardholder Data*

3. Protect stored cardholder data.
4. Encrypt transmission of cardholder data across open, public networks.

- *Maintain a Vulnerability Management Program*

5. Protect all systems against a malware and regularly update anti-virus software or programs.
6. Develop and maintain secure systems and applications.

- *Implement Strong Access Control Measures*

7. Restrict access to cardholder data by business need-to-know.
8. Identify and authenticate access to system components. E.g. use IAM or services such as Auth0
9. Restrict physical access to cardholder data. E.g. copies of credit card records

- *Regularly Monitor and Test Networks*

10. Track and monitor all access to network resources and cardholder data. E.g. CloudTrail, CloudWatch and other logging tools etc. or use a 3rd-party service to perform monitoring.
11. Regularly test security systems and processes. E.g. Pentesting, simulated phishing etc.

- *Maintain an Information Security Policy*

12. Maintain a policy that addresses information security for all personnel.

SAS70

- Statement of Auditing Standards No. 70 SOC 1
- Service Organization Controls - accounting standards. FISMA
- Federal Information Security Modernization Act.

FIPS 140-2 is a U.S government computer security standard used to approve cryptographic modules.

- Rated from Level 1 to Level 4 (highest)
- AWS CloudHSM meets Level 3.

Check out https://aws.amazon.com/compliance.

Chapter 7 Summary

AWS Shield

- Free service that protects all AWS customers on *Elastic Load Balancers*, *CloudFront* and *Route53*.
- Protects against SYN/UDP Floods, Reflection Attacks and other layer 3/4 attacks.
- Advanced Shield protects you against larger and more sophisticated attacks - $3,000 per month cost.

AWS Advanced Shield

- Always-on, flow-based monitoring of network traffic and active application monitoring.
- Real-time notifications of DDoS attacks.
- DDoS 24/7 Response Team to manage and mitigate application-layer DDoS attacks.
- Protects AWS bill against higher fees due to ELB, CF, R53 usage spikes during DDoS.

DDoS

- Remember technology that can be used to mitigate a DDoS: *CloudFront, Route53, ELBs, WAFs, Autoscaling, CloudWatch*.

EC2 has been hacked

- Stop instance immediately.
- Take snapshot of EBS volume.
- Deploy instance in an isolated environment. Isolated VPC, no internet access - ideally private subnet.
- Access instance via. forensic workstation.
- Read through logs to figure out how (Windows Event Logs).

Keys leaked on Github

- Delete key from user's account + generate a new key.

AWS Certificate Manager

- SSL Certificates renew automatically, provided you purchase a domain name from Route53 and it's not for a Route53 Private Hosted Zone.
- Use Amazon SSL Certificates with both Load Balancers and CloudFront.
- You cannot export the Amazon SSL Certificates (so you can only use it on AWS infrastructure.)

Perfect Forward Secrecy

- Someone who compromises your private key cannot use it to decrypt past traffic.
- **ECDHE** TLS cipher is needed -> by default choose `2016-08` Security Policy.

API Gateway Throttling

- Prevents your API from being overwhelmed by too many requests.
- When request submissions exceed steady-state request rate and burst limits, API Gateway fails all exceeding requests and returns a `429 Too Many Requests` response to the client.
- Steady-state rate = 10,000 requests per second.
- Account-level rate limit and burst limit can be increased upon request.

API Gateway Caching

- Cache your endpoint's response - reduce number of calls to the endpoint and improve latency of responses from endpoint.
- API Gateway caches response for a specified TTL (time-to-live) period in seconds.
- Default TTL = 300 seconds | Maximum TTL = 3600 seconds | TTL = 0 disabled caching.

AWS SSM Run Command

- Commands applied to a group of instances based on AWS instance tags or by selecting manually.
- SSM Agent needs to be installed (installed by default on some AMIs).
- Systems Manager Document defines the commands and parameters run.
- Works on-premise systems and EC2 instances.

Additional Topics

AWS Athena

Athena is an interactive query service, allowing you to query and analyse data in S3 using SQL.

- Serverless, pay per query / per TB scanned.
- No complex Extract/Transform/Load (ETL) processes.
- Works with structured, semi-structured, unstructured data and many files types such as .csv .json.

What can Athena be used for?

- Run queries on log files stored in S3 e.g. ELB logs, S3 access logs, CloudTrail logs etc.
- Generate business reports on data stored in S3.
- Analyse AWS Cost and Usage reports.

Querying CloudTrail data stored in S3 using Athena

1. Create an empty database.
2. Create tables inside the database.
3. Example `cloudtrail_logs` table:

```
CREATE EXTERNAL TABLE cloudtrail_logs (
eventversion STRING,
useridentity STRUCT<
               type:STRING,
               principalid:STRING,
               arn:STRING,
               accountid:STRING,
               invokedby:STRING,
               accesskeyid:STRING,
               userName:STRING,
sessioncontext:STRUCT<
attributes:STRUCT<
               mfaauthenticated:STRING,
               creationdate:STRING>,
sessionissuer:STRUCT<
               type:STRING,
               principalId:STRING,
               arn:STRING,
               accountId:STRING,
               userName:STRING>>>,
eventtime STRING,
eventsource STRING,
eventname STRING,
awsregion STRING,
sourceipaddress STRING,
useragent STRING,
errorcode STRING,
errormessage STRING,
requestparameters STRING,
responseelements STRING,
additionaleventdata STRING,
requestid STRING,
eventid STRING,
resources ARRAY<STRUCT<
               ARN:STRING,
               accountId:STRING,
               type:STRING>>,
eventtype STRING,
apiversion STRING,
readonly STRING,
recipientaccountid STRING,
serviceeventdetails STRING,
sharedeventid STRING,
vpcendpointid STRING
)
ROW FORMAT SERDE 'com.amazon.emr.hive.serde.CloudTrailSerde'
STORED AS INPUTFORMAT 'com.amazon.emr.cloudtrail.CloudTrailInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://mycloudtrailbucket-faye/AWSLogs/757250003982/';
```

4. Now you can perform a query on the specified CloudTrail bucket using Athena:

```
SELECT
 useridentity.arn,
 eventname,
 sourceipaddress,
 eventtime
FROM cloudtrail_logs
LIMIT 100;
```

Macie

Macie helps protect sensitive data (e.g. PII) in S3.

- Uses Machine Learning and Natural Language Processing (NLP) to discover, classify and protect sensitive data stored in S3.
- Includes dashboard visualisation, reporting and alerts.
- Great for PCI-DSS compliance and preventing identity theft.

Personally Identifiable Information (PII)

- Personal data used to establish an individual's identity.

- Exploited for identity theft and financial fraud.
- Includes: Home address, email address, SSN, Passport No., Drivers License, DOB, phone number, bank account number, credit card number.

How does Macie work? It classifies your data by four different domains:

1. By **Content Type**: JSON, PDF, Excel, TAR, ZIP, source code, XML.
2. By **Theme**: Amex, Visa, Mastercard credit card keywords, banking or financial keywords, hacker and web exploit words.
3. By **File Extension**: .bin .c .bat .exe .html .sql.
4. By **Regular Expression**: aws_secret_key, RSA Private Key, SWIFT Codes, Cisco Router Config.

How can Macie protect your data?

1. Analyse and classify data.
2. Dashboards, alerts and reports on the prescence of PII.
3. Gives visibility on how the data is being accessed.
4. Analyse CloudTrail logs and report on suspicious API activity.

Using Macie:

- Macie can only monitor S3 buckets within same region.
- Macie uses a service-role `AWSServiceRoleForAmazonMacie` which cover mainly permissions for CloudTrail (creating/reading logs) and S3 (creating/deleting buckets and objects).
- An S3 CloudTrail bucket will be created to capture all data events associated with Macie.
- Select AWS ACCOUNT ID to integrate Macie with -> Select ALL BUCKETS -> START CLASSIFICATION.
- Query S3 bucket/object properties to find PII / sensitive data.

GuardDuty

GuardDuty is a threat detection service which uses Machine Learning to continuously monitor for malicious behaviour, such as:

- Unusual API calls, calls from a known malicious IP.
- Attempts to disable CloudTrail logging.
- Unauthorized deployments.
- Compromised EC2 instances.

- Recon by would-be attackers.
- Port scanning, failed logins.

Features:

- Alerts in GuardDuty console and to CloudWatch events.
- Receive feeds from 3rd-parties such as Crowdstrike and AWS Security, obtaining info about malicious domains / known IPs.
- Monitors CloudTrail Logs for all API activity, VPC Flow Logs, DNS Logs (by default all EC2s use AWS DNS - GuardDuty records all requests to/from AWS DNS from your EC2 instances)
- Centralise threat-detection across multiple AWS accounts.
- Automated response: GuardDuty detects compromised instance -> trigger CloudWatch Events -> trigger Lambda e.g. to isolate EC2 by updating security group + take snapshot.
- Machine Learning and anomoly detection.

Setup:

- Takes 7-14 days to establish a baseline - what is normal behaviour in your account?
- Once active, findings will appear on GuardDuty console and in CloudWatch ONLY if GuardDuty detects behaviour it considers a threat.
- 30 days free. Charged based on *quantity of CloudTrail events* and *volume of DNS and VPC Flow Logs*.

Secrets Manager

AWS Secrets Manager is a service which securely stores, encrypts and rotates DB credentials and other secrets.

- Encryption in-transit and at-rest using KMS.
- Automatically rotates credentials.
- Apply fine-grained access control using IAM policies.
- Your application makes an API call to Secrets Manager to retrieve the secret programatically.
- Reduces the risk of credentials being compromised.

What credentials can I store in secrets manager?

- RDS credentials (most common use-case)

- Credentials for non-RDS databases (e.g DynamoDB)
- Any other type of secrets, as long as you can store it as a *key:value pair* (SSH keys, API keys)

Secrets Manager vs. Parameter Store

- Secrets Manager (mainly for DB credentials / keyvalue pairs)
    - Database credentials, API/SSH keys.
    - Built-in integration with RDS: MySQL, PostgreSQL, Aurora.
    - Built-in rotation of RDS secrets, support for non-RDS using Lambda.
    - $0.40/secret per month | $0.05 per 10,000 API calls.
- Parameter Store
    - Passwords, database strings, license codes, parameter values, config data.
    - User-defined parameters.
    - Values may be cleartext or encrypted (Secure String Parameter).
    - No additional charge.
    - Integrated with AWS Systems Manager.

Secrets Manager - Automatic secrets rotation

- Setting: `Enable Automatic Rotation`
    - **WARNING: If you enable rotation, Secrets Manager immediately rotates the secret once to test the configuration. If your apps are using embedded (hardcoded) credentials, do not enable rotation as it will break your app**.
    - This is the recommended setting if your apps are NOT using embedded (hardcoded) credentials
- Setting: `Disable automatic rotation`
    - This is the recommended setting if your apps are using embedded (hardcoded) credentials.
    - Ensure that your apps are updated to retrieve credentials from Secrets Manager.

There is a 7-day minimum waiting period for deleting a secret.

Simple Email Service (SES)

AWS Simple Email Service is a cloud-based email service, which supports both sending/receiving emails from your apps.

- Can be used to send marketing emails, transaction emails and email notifications from your apps.
- SES can be accessed using a standard SMTP (Simple Mail Transfer Protocol) interface OR via. the SES API, to allow you to integrate with existing applications.
- All connections to the SMTP endpoint must be encrypted in-transit using TLS.
- There are several different SMTP endpoints you can use when configuring connections into SES.
  See https://docs.aws.amazon.com/ses/latest/DeveloperGuide/smtp-connect.html.
  - Your EC2 instances will need to connect to these endpoints if it wants to send emails using SES.

Configuring Access to SES for EC2 instances

- Configure the Security Group associated with your EC2 instances to allow connections TO the SES SMTP endpoint.
- Default port is `port 25`, but *EC2 throttles email traffic over port 25* (you can raise request to increase limit).
- However, avoid timeouts by using either `port 587` or `port 2587` instead.

Security Hub

AWS Security Hub is

1. *A central hub for Security Alerts*: a single place to manage / aggregate findings and alerts from key AWS security services. Centralised dashboard.
2. *Automated checks*:
   - PCI-DSS (payment card industry)
   - CIS (Center for Internet Security)
3. *Ongoing security audit for all AWS accounts*

Security Hub integrates and with:

- GuardDuty (threat detection).
- Macie (PII and secrets in S3 buckets).
- Inspection (checks for CVEs).
- IAM Access Analyzer (scans IAM policies attached to resources that provide external access).
- AWS Firewall Manager (centrally manage WAF and SGs across multiple AWS accounts)

- External 3rd-party tools.
    - Pulling in findings from CheckPoint, F5, AlertLogic etc.
- CloudWatch (CloudWatch Events, to trigger a Lambda/SIEM/3rd-party system to take action).

Additional notes:

- Security Hub uses AWS Config for some CIS Benchmark checks (takes 12 hours from enabling AWS Config to notice changes).
- Takes time to pull info from GuardDuty / Inspector etc.

Network Packet Inspection in AWS

Network Packet Inspection involves inspecting packet headers and data content of the packet (known as DPI - Deep Packet Inspection).

- Filters non-compliant protocols, viruses, spam, intrusions.
- Takes action by blocking, re-routing or logging.
- IDS/IPS combined with a traditional firewall.

None of the following AWS services provide Network Packet Inspection

- VPC Flow Logs, AWS WAF, host-based firewalls like iptables and Windows Firewall.

What to do?

- Use a 3rd-party solution for Network Packet Inspections / IDS/IPS.
- Install 3rd-party software on your EC2 instance.
- Search the AWS Marketplace (Alert Logic, Trend Micro, McAfee) https://aws.amazon.com/marketplace.

Active Directory Federation with AWS

AD Federation with AWS

- AWS allows federated sign-in to AWS using Active Directory credentials.
- Minimises the admin overhead by leveraging existing user accounts, passwords, password policies and groups.
- Provides SSO for users.

- *Great for companies that have an existing Active Directory Domain and you have corporate users who have AD accounts - you don't want to recreate those accounts in AWS.*

Active Directory Terminology

- **ADFS (Active Directory Federation Services)**: Provides Single Sign-On (SSO) and acts as an Identity Broker between the Active Directory Domain in your data centre and AWS.
    - Runs directly inside your data centre.
- **SAML 2.0 (Security Assertion Markup Langauge)**: An open standard for exchanging identity and security information between identity providers and applications. Enables SSO for AWS accounts.
    - Allows users to use the AWS API and sign into AWS console using their corporate accounts.

Establishing AD Federation with AWS: **2-way trust**:

- Within AWS, you need to configure *ADFS = Trusted Identity Provider*.
    - "You need to tell AWS, to trust ADFS to provide your users' identities."
- Within ADFS, you need to configure *AWS = Trusted Relying Party*.
    - "You need to tell ADFS, to trust AWS to consume your users' identities."

Using ADFS to sign-in to AWS Console (after 2-way trust is configured0):

1. User logs into ADFS using a special ADFS sign-in webpage. They provide their company AD/corporate username and password.
2. ADFS will authenticate against the company Active Directory.
3. ADFS sends back authentication response in the form of a SAML token.
4. User's browser sends SAML token to AWS sign-in endpoint.
5. AWS sign-in endpoint makes an `STS AssumeRoleWithSAML` request to request temporary credentials to AWS console and STS returns temporary credentials.
6. AWS sign-in endpoint redirects user to the AWS console and the user will use temporary credentials to get access to the console.

AWS Artifact

AWS Artifact is a central resource for compliance and security related information.

- Download AWS security and compliance documents.
- ISO 127001 certifications, PCI-DSS docs, SOC (Service Organizational Control) reports.

AWS Artifact enables you to:

- Demonstrate compliance to regulators.
- Evaluate your own cloud architecture.
- Assess the effectiveness of your company's internal controls.
- *Example: if your company plans to launch a new credit card, AWS Artifact can help with the above*

Additional Resources for Exam Preparation

AWS White Papers: https://aws.amazon.com/whitepapers

- There is a whole section dedicated to security.
- Important whitepapers to help pass Security Cert Exam: *KMS Best Practices, KMS Cryptographic Details, DDoS Best Practices, Logging in AWS, Well-Architected Framework - Security Pillar*
- Tip: Read the intro, the summary and choose the sections that you are weaker in (don't need to read the whole thing).

re:Invent Videos

- Great videos for exam prep
    - KMS Best Practices, AWS Encryption Deepdive, DDoS Best Practices
- Become an IAM Policy Master, VPC Fundamentals & Connectivity Options, Advanced Security Best Practices Masterclass.

Read FAQs on main security services.

- https://aws.amazon.com/faqs/
- Read all the FAQs under the *Security, Identity and Compliance* heading.
- Best to focus on are KMS and IAM FAQs.

Free Practice Questions

AWS Certification Quiz Shows

- Video with a focus on Security Specialty Exam.
- Explains how to approach the exam.
- Video 1: https://www.twitch.tv/aws/video/467770461
- Video 2: https://www.aws.training/Details/Video?id=37283
- Video 3: https://www.aws.training/Details/Video?id=37284

- Video 4: https://www.aws.training/Details/Video?id=37293

Other resources

- Mock exams in AWS Training ($20 each): semi-realistic questions to what you will see in the real exam.
- Free course: https://www.aws.training/Details/eLearning?id=49720

Other tips

- Do practise tests.
- Go through the explanation for BOTH incorrect and correct answers and study the AWS documentation associate with the answer to make sure you understand the subject well.

Troubleshooting Scenarios

Troubleshooting Monitoring & Alerting - CloudWatch

Common issue: Does the IAM user/role or AWS Service have correct permissions to allow them to read/write?

Example: Issue with IAM user to reading CloudWatch dashboard

- Policy to allow user to read dashboard

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "autoscaling:Describe*",    // view EC2 autoscaling
                "cloudwatch:Describe*",     // view CloudWatch metrics
                "cloudwatch:Get*",
                "cloudwatch:List*",
                "logs:Get*",                // view CloudWatch logs data
                "logs:Describe*",
                "sns:Get*",                 // view alarm-related SNS data
                "sns:List*"
            ]
        }
    ]
}
```

Example: Issue with EC2 sending logs to CloudWatch

- Is the CloudWatch agent installed on EC2?
- Is the CloudWatch agent running on EC2?
- Does the instance role have permissions to write to CloudWatch Logs?

Example: Unauthorised user able to create EC2 instances -> CloudTrail Logs API calls -> Send Event into CloudWatch Events -> CloudWatch Event triggers Lambda -> Lambda terminates unauthorised instances.

- Check that CloudWatch Events has permission to invoke the event target (Lambda)
- Check the Lambda's `Execution Role` (IAM role associated with Lambda upon execution) has permissions to terminate EC2

Exam tips

- Always check that IAM users have the correct permissions to allow them to do what they need to do.
- CloudWatch Logs require an agent to be *installed* AND *running* on your EC2 instance.
- For CloudWatch Events, make sure the *Event Target* (Lambda, SNS, SQS, Kinesis) has the correct permissions to take whatever action it needs to e.g. *does the Lambda execution role include permissions to terminate an EC2?*

Lambda permissions

Lambda *Function Policy*: defines which AWS resources are allowed to invoke your function. Lambda *Execution Role*: defines which AWS resources your Lambda function can access, and what actions can be taken against those AWS resources.

Troubleshooting CloudTrail Logging

Common issue: *Logging not working* (CloudTrail logs not appearing in S3)

- Is CloudTrail enabled?
- Have you provided the correct S3 bucket name?
- Is the S3 Bucket Policy / S3 Access Control List correct?

Common issue: *Added costs*

- S3 Data Events and Lambda Data Events are high volume / high cost.

  - Data Events are NOT enabled by default.
  - S3 Data Events: record S3 object-level API activity (e.g `GetObject` and `PutObject`) for individual or ALL buckets.
  - Lambda Data Events: record invoke API operations for individual or ALL functions.

Common issue: *Auditor is not able to access logs*

- Does the Auditor's account read access to CloudTrail?
  - By default, normal IAM users don't have access to CloudTrail logs. Explicit access is needed.
  - `AWSCloudTrailReadOnlyAccess` IAM Policy will allow access to CloudTrail logs.

Troubleshooting Secure Network Infrastructure - VPC

Troubleshooting VPCs

- Check routing tables, Security Groups, NACLs.
  - *Public traffic -> Public subnets*: make sure routing table is routing internet traffic to the INTERNET GATEWAY.
    - FLOW: Internet traffic -> VPC Internet Gateway -> VPC Router -> Routing Table -> NACL -> Security Group -> Public subnet.
  - *VPN traffic -> Private subnets*: make sure routing table is routing any traffic to your own datacenter through the VIRTUAL PRIVATE.
    - FLOW: VPN traffic -> VPC Virtual Private Gateway -> VPC Router -> Routing Table -> NACL -> Private Subnet -> Security Group -> instance
  - Check that *Security Groups* and *Network Access Control Lists* are permitting the traffic.
- Internet access - NAT Gateway, Internet Gateway.
- Check VPC Flow Logs to view `ALLOW` or `DENY` messages.
  - `DENY` messages should give you a clue as to where the problem might be.

Exam tips

- NACLs are stateless: you need to configure both INBOUND and OUTBOUND rules.
- Security Groups deny by default, use NACL to explicitly deny.

- If you are peering 2 VPCs, remember to configure routing tables in both VPCs so they know how to route traffic to each other.
- Problems with internet access: make sure you have configured your Routing Tables correctly, to use either a NAT Gateway OR Internet Gateway.

Troubleshooting Authentication & Authorization

Authentication & Authorization issues: "Giving users the ability to access resources they need to perform their job, no more and no less".

Common issues with Conflicting Policies

- AWS authZ/authN takes a least privilege approach: all actions are DENY BY DEFAULT. You need to EXPLICITLY ALLOW permissions for actions you want users to perform.
- Explicit deny will always override an allow.
- With multiple policies in play e.g. IAM Policy, S3 Bucket Policy, S3 ACL, Key Policy, an action is only allowed if NO METHOD EXPLICTLY DENIES and AT LEAST ONE METHOD EXPLICITLY ALLOWS access.
- If you are using AWS Organisations: check if there is a PERMISSIONS BOUNDARY preventing the action.

Conflicting Policy Example: S3 Bucket Policy allowing all S3 actions for IAM user `FAYE` on all S3 resources, conflicting with an IAM Policy.

```
// S3 Bucket Policy for LOG S3 bucket
{
    "Version": "2012-10-17",
    "Statement": [
        "Effect":"Allow",
        "Action":"s3:*",
        "Principal": {
            "AWS": "arn:aws:iam::11223344:user/brian"
        },
        "Resource":"*"
    ]
}

// IAM Policy attached to user brian
{
    "Statement": [
        {
            "Sid":"AllowS3AccessToMyOwnBucket",
            "Effect":"Allow",
            "Action":"s3:*",
            "Resource": [
                "arn:aws:s3:::mybucket/*"
            ]
        }
        // This DENY statement overrides ALLOW statement in the S3 Bucket Policy
        {
            "Sid":"DenyS3AccessToLogsBucket",
            "Effect":"Deny",
            "Action":"s3:*",
            "Resource": [
                "arn:aws:s3::::*log/*"
            ]
        }
    ]
}
```

Troubleshooting Identity Federation

- Use the correct API for the job
- Authenticated by a Web Identity Provider (Facebook etc.): `STS:AssumeRoleWithWebIdentity` API call.
- Authenticated by a SAML Compliant ID Provider (Active Directory etc.): `STS:AssumeRoleWithSAML` API call.
- Authenticated by AWS: `STS:AssumeRole` API call.

Read more about Policy Evaluation Logic (worth reading): https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_evaluation-logic.html

Troubleshooting Cross Account Access with STS:AssumeRole API

This definitely comes up in the exam.

Example: Dev users ReadOnly access to a Prod S3 bucket (cross-account access).

1. Create Prod IAM Role with ReadOnly access to Prod S3 bucket.
2. Allow Dev IAM User to assume the above role via. Trusted Relationship statement.

Common issue: *Check external account (Dev account) has permission to call `STS:AssumeRole`*

```
// Dev IAM Policy attached to Dev IAM User - assume Prod IAM Role
"Statement":[
    {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "arn:aws:iam::PRODUCTION_ACCOUNT-ID:role/ROLE-NAME"
    }
]
```

Common issue: *Check the external account is trusted AND has permission to perform the action you are attempting - Prod Account, Role.*

```
// Prod IAM Role - Add Trusted Relationship statement / Configure Dev account as a
Trusted Entity + give permission to perform the STS:AssumeRole action.
"Version": "2012-10-17",
"Statement":[
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": [
                "arn:aws:iam::DEVELOPMENT_ACCOUNT-ID:root"
            ]
        },
        "Action": "sts:AssumeRole"
    }
]
```

Example: Cross-account KMS access.

Common issue: *Key Policy needs to trust external account*.

1. Go to Prod Account KMS -> your CMK which you want to allow external access.
2. Select "Other AWS Accounts" add AWS account ID.

Common issue: *External account needs IAM Policy allowing users to run specific API calls related to resource (CMK in this case)*

```
// Dev IAM Policy
"Statement":[
    {
        "Sid": "AllowUseOfCMKInAccount444455556666",
        "Effect": "Allow",
        "Action":[
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:DescribeKey"
        ],
        "Resource": "arn:aws:kms:us-west-2:444455556666:key/1a2b3c"
    }
]
```

Exam Tips:

- *For cross-account access to S3*:
  i. Check that the IAM Policy in EXTERNAL account (Dev) needs to allow the user to call `STS:AssumeRole`
  ii. Check that the IAM Policy in TRUSTING account (Prod) needs to allow the action.
- *For cross-account access to KMS*:
  i. Check that you have configured the `Key Policy` to allow access to the EXTERNAL account in the TRUSTED account.

   ii. Check that you have configured the IAM Policy in the EXTERNAL account to take KMS actions on the TRUSTED account.

- THE TWO MAIN IDEAS FOR CROSS-ACCOUNT ACCESS:
   i. Enable access within the TRUSTED account sharing the resource.
   ii. IAM Policy in the EXTERNAL account, allowing actions by a role/user in the account.

Troubleshooting Lambda Access

Example: You want Lambda executions to be logged in CloudWatch. Common issue: *Lambda require permissions to write to CloudWatch Logs*

- This is defined by the Lambda EXECUTION ROLE, similar to EC2 Service Role (NOT Function Policy).
- NOTE: *Lambda Execution Role* defines what the Lambda fn can do.
- NOTE: *Function Policy* defines which services can invoke the Lambda fn.

Example: You want a Lambda fn to access a RDS database Common issue: *Lambda requires permissions to access Secrets Manager (RDS db credentials held there)*

- This is defined by the Lambda EXECUTION ROLE.

Example: You want CloudTrail to report certain type of Events into CloudWatch Events, then invoke Lambda Common issue: *Does the Lambda Function Policy allow CloudWatch Events to invoke the Lambda?*

Exam tips:

- *Lambda cannot perform an action*
    - E.g. write to S3, log to CloudWatch, Terminate Instances, use a CMK, use Secrets Manager
    - Check the LAMBDA EXECUTION ROLE allows the actions.
- *Lambda cannot be invoked by service*
    - E.g. CloudWatch Event invoking Lambda fn.
    - Check the LAMBDA FUNCTION POLICY allows the service.
- Remember that some services have their own resource-based policies which will impact access to the resource
    - E.g S3 Bucket Policy, KMS Key Policies etc.

Troubleshooting Access To CMKs in KMS

Example: Accessing a CMK in KMS Common issue: *Check that IAM user, group or role has permissions for the action they are attempting* * Access to use KMS Customer Master Keys is defined by:

1. **IAM Policy** attached to User, Group or Role.
   - Defines actions such as `kms:ListKeys`, `kms:Encrypt`, `kms:Decrypt`.
2. **CMK Key Policy**
   - Defines `Key Admins`, `Key Users`, `trusted external accounts`.

Summary

Networks Glossary

Dynamic Host Configuration Protocol (DHCP): assigns IP addresses dynamically to hosts in a PRIVATE network.

Network Address Translation (NAT): method of rewriting packets to allow multiple devices on a private network to share a single PUBLIC IP address.

Route Table: contains a set of rules (routes) that are used to determine where network traffic from your subnet or gateway is directed. It allows subnets to talk to each other. Every AWS subnet provisioned will automatically be attached to your default/main route table. Ideally, you should create a separate route table that is internet accessible rather than use your default/main route table as every new subnet being provisioned will be associated with the default/main route table, thus becoming internet accessible.

Subnets: The purpose of subnetting is to help relieve network congestion. If you have an excessive amount of traffic flow across your network, then that traffic can cause your network to run slowly. When you subnet your network, most of the network traffic will be isolated to the subnet in which it originated. Ideally, your subnet structure should mimic your network's geographic structure. Other benefits of subnetting include routing efficiency, easier network management and improving network security.

Chapter 2 - IAM, S3 and Security Policies

Resetting Root Users

- CHANGE root user password / strong password policy.
- DELETE 2FA then re-enable 2FA.

- ROTATE, then DELETE Root and IAM user access keys.
- DELETE IAM users that have potentially been compromised.
- DELETE AWS resources you didn't create

Resource vs. Identity policies

- **Resource Policies**: S3 Bucket Policy, CMK Key Policy.
- **Identity Policies**: IAM Policy attached to IAM entities.

IAM Role

- A role **Trust Policy** is a policy that defines the principals that you TRUST TO ASSUME the role. Principals can be users, roles, accounts, services.
- A role **Permissions Policy** is a policy that defines the ACTIONS/RESOURCES THE ROLE CAN PERFORM/USE.

S3 Bucket Policy vs. Bucket ACL

- **Bucket Policy**: for controlling access to Buckets (recommended approach, as controlling multiple ACLs for multiple objects is difficult vs. 1 Bucket Policy).
- **Bucket ACL**: for controlling access to Buckets AND Objects, or need to exceed 20kb policy max size.

S3 Encryption: Client-side and server-side

- **Encrypting S3 object metadata**:
  - *SERVER-SIDE ENCRYPTION ONLY APPLIES TO OBJECT DATA, NOT OBJECT METADATA.*
  - To encrypt metadata, store it in a DynamoDB table and turn on encryption during table creation time.
- **S3 Bucket Encryption**: All new S3 objects are encrypted when they are stored in the bucket.
- **S3 Client-Side Encryption**: is possible using AWS SDKs, by encrypting an S3 object using a KMS CMK or a Master Key you store in your own application before uploading the S3 object to an S3 bucket.
- **SSE-C**: allows you to set your own external encryption keys.
  - S3 rejects non-HTTPS requests when using SSE-C.
  - If you lose the external encryption key, you will lose access to the object (AWS cannot assist).
- **SSE-KMS**: encryption using KMS service - AWS-managed CMK or Customer-managed CMK.
- **SSE-S3**: encryption using S3-managed encryption key.

S3 Bucket Policy / ACL / IAM conflicts:

- **Explicit Deny Overrides**: An EXPLICIT DENY will always override any ALLOW.
- **Policy Conflicts**: Whenever an AWS principal (user, group or role) issues a request to S3, the authorization decision depends on the union of all the IAM policies, S3 bucket policies and S3 ACLs that apply.
- **Policy Conflict flow**: (1) DENY by default (2) If policy has EXPLICIT DENY = DENY (3) If policy has ALLOW = ALLOW

S3 Cross-Region Replication (CRR): replicate objects across S3 buckets in different AWS regions

- **CRR replicates**: NEW objects (*encrypted with SSE-S3/SSE-KMS or unencrypted*), metadata, ACL updates, tags.
- **CRR CANNOT replicate**: objects before CRR, objects encrypted by SSE-C, objects which bucket owner does NOT have permissions, object deletes of a specific version.

S3 Bucket Cross-Region Replicate with Cross-Accounts:

- **Audit use-case**: (1) CT logs acct-XYZ (2) CRR turned on to replicate CT logs to acct-Audit (3) acct-XYZ can only replicate logs to acct-Audit but NOT read/write to acct-Audit.
- **Permissions**: IAM role must have permissions to replicate objects in the destination bucket.
- **CRR Config**: You can optionally direct S3 to change ownership of object replicates to the AWS account that owns the destination bucket.

S3 Bucket CRR - encrypting replica objects in destination bucket using SSE-KMS: what CRR config is required?

- CONFIG #1: The CMK referenced by `ReplicaKmsKeyID` needs to be added to CRR configuration.
- CONFIG #2: Ensure the CMK is in the same region as the destination bucket.

S3 bucket access via. CloudFront - using Origin Access Identity (OAI)

- **CloudFront Origin Access Identity** is a virtual identity used to give a CF distribution permission to fetch a private object from an S3 origin on behalf of end-users. All direct access by using S3 URLs will be denied.
- Steps to enable: (1) Create an OAI in CloudFront + turn on `Restrict Bucket Access` (2) Update S3 bucket permissions: turn on `Grant Read Permissions on Bucket` OR change permissions manually in S3 bucket to allow OAI access.

- OAI principal with bucket access should be `arn:aws:iam::cloudfront:user/CloudFront Origin Access Identity EAFXXXXXID`

S3 bucket access PRIVATE access - using VPC Endpoints

- **S3 Bucket Policy** with `aws:SourceVpce` enforces access to S3 bucket only via. the specified VPC Endpoint.
- All traffic stays within the VPC, hence remains private.

S3 object access via. Pre-signed URL - for a temporary S3 object access using your own credentials

- Presign URL with 300s expiry: `$ aws s3 presign s3://examplepresigned/hello.txt --expires-in 300`
- Example response: https://examplepresigned.s3.amazonaws.com/OBJECT.txt?AWSACcessKeyId=XXX&Expires=XXX&x-amz-security-token=XXX&Signature=XXX

S3 object encryption - enforce object encryption (AES256) in a bucket

- **S3 Bucket Policy** can be used to DENY `s3:PutObject` if object is not encrypted with `AES_256`.

AWS Security Token Service (STS): grants users limited and temporary access to AWS resources. Key terms:

- **Identity Federation** is a system of trust between two parties for the purpose of authenticating users and conveying information/attributes needed to authorize their access to resources.
- **Identity Store / Identity Provider (IdP)** is responsible for user authentication and stores identities (AD, FB, Google).
- **Service Provider (SP)** is responsible for controlling access to resources.
- **Identity Broker** is the service that allows you to take an identity from A and federate it to B.

AWS STS authentication steps:

1. **As the Identity/User**, authenticate against the **Identity Store / Provider** (Okta, AD, FB, Google) using user/pass.
2. **As the Identity Broker**, authenticate against STS using **sts:GetFederationToken** to obtain a temp STS token.
3. **As the Application**, authenticate against AWS service with temp STS token to obtain access to the requested resource.

Web Identity Federation with Amazon Cognito (web and mobile app authN/authZ):

- **Amazon Cognito**: An Identity Broker to connect a WebApp to users from Identity Store/Provider like Facebook.
- **Cognifo benefits**: No need for mobile app to embed AWS credentials locally on device + provides user with seamless experience.
- **Cognito User Pools** are for authentication. With a User Pool, your app users can sign-in through the User Pool OR federate through a 3rd-party identity provider (IdP).
    - **User Pool Attributes** are pieces of info that help you identify individual users, such as *name, email, mobile*.
- **Cognito Identity Pools** are for authorization. Use Identity Pools to authorize your users (sourced from User Pools, FB, Google etc.) to different AWS services. You can also generate temporary AWS credentials for unauthenticated users.
- **Access to API Gateway via. Cognito User Pools as authorizer**: You can use a Cognito User Pool to control access to your APIs in API Gateway as an alternative to IAM roles and policies or Lambda authorizers.
    1. Create a Cognito User Pool -> Create an API Gateway authorizer w/ the chosen User Pool -> Enable the authorizer on selected API methods.
    2. User obtains an identity token from the User Pool after authentication, then the identity token is passed to the `Authorization` header in the API request.

Glacier Vault Lock: low-cost storage service for data archiving and long-term backup

- **Archives** is a single file or multiple files stored in .tar/.zip.
- **Vault** is a container which stores one or more archives.
- **Vault ACCESS Policy** is for implementing ACCESS CONTROL rather than a Lock Policy which is compliance-related.
- **Vault LOCK Policy** is used to configure **WRITE ONCE READ MANY (WORM)** archives / create data retention policies.
- **Vault Locking steps**:
    1. CREATE Vault Lock Policy.
    2. INITIATE Vault Lock (`POST lock-policy`) -> attaches policy to your Vault -> 24hrs to validate policy, otherwise the policy is detached/removed from the Vault.
    3. TEST Vault Lock: ABORT Vault Lock (`DELETE lock-policy`) to detach policy and re-attach with INITIATE Vault Lock until you fine-tune the policy.
    4. COMPLETE Vault Lock (`POST lockId`) -> Vault is now locked and policy is UNCHAGEABLE/IMMUTABLE.

- Example Vault Lock Policy: enforce archive retention for 1 year:

```
{
        "Sid": "deny-based-on-archive-age",
        "Principal": "*",
        "Effect": "DENY",
        "Action": "DeleteArchive":,
        "Resource": [
                "arn:aws:glacier:us-west-2:XXXaccountidXXX:vaults/examplevault"
        ],
        "Condition": {
                "NumericLessThan": {
                        "glacier:ArchiveAgeInDays": "365",
                }
        }
}
```

AWS Organisations: Service Control Policies (SCPs)

- **IAM entities** can be restricted at an account-level, as long as accounts are inside the AWS Organisation.
- **ROOT users** can be restricted, as long as accounts are inside an AWS Organisation.
- **Service-linked roles** are NOT restricted by SCP.
- **Master account actions** are NOT restricted by SCP.
- SCP never GRANTS permissions, only LIMITS permissions in member accounts, including each member account Root user.

IAM Credential Report is a CSV-formatted report which lists all users in accounts + status of their various credentials, including PASSWORDS, ACCESS KEYS, MFA devices (last used, rotated).

- Requires both `iam:GenerateCredentialReport` AND `iam:GetCredentialReport`.

Chapter 3 - Logging and Monitoring

CloudTrail: securing CloudTrail log files

- CloudTrail Event History is turned on by default (showing 90 days of activity). For longer-term logging, create a Trail and specify an S3 bucket to deliver events to. Trails by default log `Management Events` but not `Data Events OR Insight Events`.
- **Validate log file Integrity via. CLI**: `$ aws cloudtrail validate-logs`
- **Prevent log file UNAUTHORISED ACCESS**: IAM/S3 bucket policies to restrict access + SSE-S3/SSE-KMS encryption.

- **Prevent log file DELETION**: IAM/S3 bucket policies to restrict access + S3 MFA Delete + validate that logs haven't been deleted via. Log File Validation.
- **Ensure log file RETENTION for X-years for COMPLIANCE**: Log files are stored indefinitely. Use S3 Object Lifecycle Management to remove files after required period of time OR move files to AWS Glacier for long-term storage.
- **Receive log files from MULTIPLE-REGIONS**: Turn on `CloudTrail Multi-Region` to delivery logs from multiple regions in a single AWS account to a single S3 bucket. Any new region launched -> CloudTrail automatically creates Trail in the new region with the same settings as your original trail.
- **CloudTrail logs in S3** are encrypted by DEFAULT with SSE-S3. You can configure a different KMS CMK while creating a Trail.

CloudTrail Log File Integrity Validation when enabled:

1. CT creates a HASH for every log file that it delivers.
2. CT then creates a DIGEST FILE that references the log files for the LAST HOUR and contains a hash of each log file.
3. CT signs each DIGEST FILE using a private/public keypair (AWS-controlled) - uses SHA-256 and SHA-256 hashing w/ RSA for digital signing.
4. After delivery of digest file, you can use the PUBLIC KEY to validate the digest file.

CloudTrail: How can we be notified that a log file has been created, then validate that its not been modified?

- Lambda to compare digest file of yesterday vs. digest of same file last week -> if digest is different, trigger SNS notification.

AWS CloudWatch: real-time monitoring for resources and applications (utilisation / operational performance)

- **CW Metrics / CW Custom Metrics**: CPU utilisation, network utilisation
- **CW Alarms**: CPU > 80% utilisation = trigger CW Alarm
- **Notifications**: SNS notifications
- **CW Logs**: monitor, store and access log files from AWS services (e.g. CloudTrail) or apps/systems (EC2 kernel logs, appserver logs). CW log retention = logs are stored indefinitely by default.

AWS CloudWatch Events: delivers near real-time stream of system events that describe changes in AWS resources.

- CW Events terminology:

  - **CW Event**: An event indicates AWS resources state change, CloudTrail API calls, custom events (HTTP 403) or scheduled/periodic events.
  - **CW Event Rule**: A rule matches incoming events and routes them to one or more targets.
  - **CW Event Target**: A target processes events. Targets include Lambda, SNS Topics, SQS queues, EC2 and more.
- Steps to create a CloudWatch Events Rule that triggers on an Event:
  i. **Event Source**: choose an AWS service (e.g. AWS Config) that you want to capture events from.
  ii. **Event Type**: choose the Event Type (e.g. `Config Rules Compliance Change`). Types available depending on the chosen source AWS service.
  iii. **Event Target**: choose an AWS service as the Event Target (e.g. Lambda) and an associated IAM role with permissions for CloudWatch Events to invoke the target.

AWS Config: continuously monitors and records AWS resource configurations and allows you to automate evaluation of recorded configurations against desired configs.

- **Provides**: configuration snapshots, logs config changes of AWS resources, automated compliance checking.
- **Enables**: compliance auditing, security analysis, resource tracking (what resource we're using and where)
- **AWS Config changes**: resource configuration changes -> AWS Config invokes `List/Describe` API call -> updating config is recorded as **Configuration Items** and delivered in a **Configuration Stream** to an S3 bucket.
- **AWS Config Rules**: resource config changes -> CloudWatch Event invokes custom/managed-rule's Lambda -> Lambda returns a compliance status.
- **Use CloudTrail to gain deeper insights**: by getting an answer on "*Who made an API call to modify the configuration of this resource?*"

Root User Monitoring via. CloudWatch: setting up an alert for Root User API activity

1. Enable delivery of CloudTrail events to a CloudWatch Logs log-group.
   - A role is required for CT to perform CloudWatch API calls. Two calls are performed:
   - `CreateLogStream`: Create a CloudWatch Logs log-stream in the CloudWatch Logs log-group you specify.
   - `PutLogEvents`: Deliver CloudTrail events to the CloudWatch Logs logs-stream.

2. Select the CW log-group -> create a **CW Metric Filter** (*defines terms/patterns to look for in log-data*) using filter: `{ $.userIdentity.type = "Root" && $.userIdentity.invokedBy NOT EXISTS && $.eventType != "AwsServiceEvent" }` -> assign Metric Filter "e.g. RootAccountUsage" to Metric "e.g. RootAccountUsageCount.
3. Create a **CW Alarm**: select Metric created above -> set a threshold level e.g.`THRESHOLD >= 1` -> set an action when an alarm-state occurs e.g. send SNS notification.

AWS Inspector: automated security assessment service to improve security/compliance of applications in your AWS account

- How it works: assessment performed -> prioritised findings produced -> findings can be reviewed directly or exported as a report via. Inspector or API
- **Assessment Template** is a configuration you define your assessment run i.e. RULES PACKAGE to evaluate target with,DURATION of assessment, SNS TOPICS which Inspector sends notifications to about run-state/findings.
- Rule packages allow you to run assessments related to a specific area:
  - **Network Reachability package** help automate monitoring of AWS networks (VPS, ELBs) and identify where network access to your EC2 instances might be misconfigured.
  - **Common Vulnerabilities and Exposures** help verify whether EC2 instances are exposed to CVEs.
  - **CIS benchmarks** and **Amazon Inspector Security Best Practices** rules.

AWS Trusted Advisor (advises on cost, performance, security, fault tolerance). Example security checks:

- **Security Groups** for unrestricted access on specific ports (SSH inbound 0.0.0.0/0), overly permissive RDS SG access.
- **Credentials**: MFA on Root, IAM password policy, IAM key rotation, exposed access keys on the internet.
- **S3**: open access to S3 buckets.
- **Logging**: CloudTrail enabled

S3 storage for logs: best service for log storage

- S3 Object Lifecycle Management.
- 99.99% durability and 99.99% availability of objects over a given year.

Chapter 4 - Infrastructure Security

KMS Customer Master Keys (CMKs): a master key, used to generate/encrypt/decrypt data keys

- **Data Encryption Keys (DEKs)** are used to encrypt your actual data = **Envelope Encryption**.
- **7-30 day waiting period** before you can delete CMKs.
- `CMK Key ADMIN` **CMK administrative actions**: `CreateKey, EnableKey, DescribeKey (get CMK metadata)` and more.
- `CMK Key USER` **CMK cryptographic actions**: `Encrypt, Decrypt, GenerateDataKey (create Data Key that is encrypted with a specified CMK).`
- **FIPS 140-2** is supported by KMS.
- **CMKs can NEVER be exported**.
- **CMKs are REGION specific**: CMKs can only be used within the same region.

KMS: Custom Key Store

- **KMS CMKs you create** by DEFAULT are generated and stored/protected by HSMs that are FIPS 140-2 Level 2 compliant.
- **KMS Custom Key Store** is a storage/protection for a KMS CMK by an AWS CloudHSM cluster, which is FIPS 140-2 Level 3 compliant. Your CMKs never leave the CloudHSM instances.
- **All KMS operations** on CMKs in a Custom Key Store are only performed in your HSMs.
- **Integration with AWS SDK/Encryption SDK and AWS services** is available to applications that use the Custom Key Store.

KMS: Create a Customer-managed CMK with Imported Key Material

1. Create **symmetric CMK** with NO key material, where material origin = EXTERNAL (non-AWS generated).
2. Download an AWS **Wrapping Key** (public key) as `PublicKey.bin` and an Import Token `ImportTokenXXX`.
3. Use `$ openssl rand 32` (random 32bit string) to generate your own key material
4. Encrypt the key material with the Wrapping Key (public key).
5. Upload `EncryptedKeyMaterial.bin` and `ImportTokenXXX` to the customer-managed CMK.

KMS: Considerations of using Imported Key Material

- `EncryptedKeyMaterial` and `ImportTokenXXX` CANNOT be used twice - they are single use only.

- **Automatic Key Rotation** CANNOT be done for CMK w/ Imported Key Material. Same applies to Asymmetric CMKs and Custom Key Stores backed by CloudHSM.
- **Manual Key Rotation** CAN be done, by repeating the process of creating a new CMK w/ new Imported Key Material
- **Key Deletion** CAN be done IMMEDIATELY, by deleting the Key Material.

KMS Key Rotation

- **AWS-Owned CMKs**: *CMKs that belongs to AWS, not the customer. CAN'T be viewed/audited.*
  - AWS manages rotation.
  - Rotation is varied - it depends on the AWS service that creates and manages the CMK.
- **AWS-Managed CMKs**: *CMKs that belong to the customer, but managed by AWS on behalf of an AWS service integrated with KMS. CAN be viewed/audited, but CANNOT be used in cryptographic operations nor change key policies.*
  - AWS manages rotation.
  - Rotation is required and occurs every **3 YEARS**.
  - NO manual rotation.
- **Customer-Managed CMKs**: *CMKs that belong to the customer, fully managed by the customer. CAN be viewed/audited, CAN change key policies/grants, add tags and create aliases.*
  - Customer manages rotation.
  - Automatic rotation every **1 YEAR** can be enabled - **ensure CMK is not hardcoded before enabling auto-rotation**.
  - Manual rotation is possible by (1) Creating new CMK (2) Update apps/key-alias to use new CMK (3) Keep old CMK so it can decrypt old objects.
  - Deletion requires **7-30 day waiting period**.
  - Deletion CANNOT be reversed as AWS deletes the Key Material + all metadata associated with the CMK.
- **Customer-Managed CMKs w/ Imported Key Material**: *Same as above, except with Imported Key Material.*
  - Customer manages rotation.
  - NO automatic rotation is possible, as Key Material is external / not AWS-generated.
  - Manual rotation by (1) Creating new CMK (2) Update apps/key-alias to use new CMK (3) Keep old CMK.
  - Deletion can be done immediately by deleting Imported Key Material from the CMK, making it unusable.

- Deletion can be reversed by re-importing the SAME Imported Key Material.

KMS CMKs in Custom Key Store (backed by CloudHSM)

- You can create CMKs in a custom key store, where KMS will generate and store key material for the CMK in a CloudHSM Cluster that you own and manage.
- Cryptographic operations are performed in the HSMs in the cluster.
- Unsupported features: NO Asymmetric CMKs, NO CMKs with Imported Key Material, NO automatic rotation.

KMS CMKs: Key Policy use with IAM policies

- A Key Policy must explicitly allow IAM to use IAM policies to give users/roles access to the CMK.
- This is done by having an `ALLOW` statement for Principal `"AWS": "arn:aws:iam::111222333:root"` (Allow IAM in account 111222333 to use CMK).

KMS Grants are used to programmatically delegate TEMPORARY use of CMKs to other AWS principals.

- Grants only ALLOW, not deny access to a CMK.
- `create-grant` adds new grant to CMK, specifies who can use it and list of operations grantee can perform. A grant token is generated and can be passed as an argument to a KMS API.

KMS Policy Conditions - ViaService is used to ALLOW/DENY access to your CMKs according to which service the request originated from.

KMS Policy Conditions - `aws:SourceVpc` is used to enforce access to your CMKs to a specific VPC Endpoint e.g. "vpce-1234abcdf5678c90a" (VPC Endpoint ID).
KMS CMK cross-account access: enable access by

1. Change CMK Key Policy in origin account to allow a specific userARN/roleARN of destination account to have access.
2. Set up an IAM policy in destination account with explicit permission to use the CMK in the origin account.
3. Attach IAM policy to userARN/roleARN in destination account.

KMS vs. CloudHSM

- **CloudHSM**: Dedicated access to HSM that complies with government standards (FIPS) + you control keys and software that uses them (you need to do your own key management).
- **KMS**: Built on the strong protections of a HSM foundation, highly available/durable, auditable, easy integration with AWS services and applications.

KMS data key ReEncryption using CMK - required permissions?

- `kms:DescribeKey` is needed to retrieve information about the CMK, to pass to `kms:ReEncrypt` call.
- `kms:ReEncrypt` is needed to re-encrypt data keys, which were encrypted with the CMK.

EC2 security

- Importing a customer-managed key pair for SSH access:
  i. use `openssl` to generate a private-key.pem using RSA 2048bits and a public-key.pem
  ii. Go to EC2 -> Import a Key Pair -> choose your public-key. Now you can provision an EC2 instance and select the public-key.
  iii. SSH into EC2 using the private key.
  iv. Add additional logins by generating a new asymmetric keypair (type=RSA) via. `$ ssh-keygen -t rsa` -> add public-key to `~/.ssh/authorized_keys` in the EC2 -> login using private-key.
- You CANNOT use KMS with SSH for EC2 as AWS is involved in generation of KMS keys.
- You CAN use CloudHSM with SSH for EC2 because you can EXPORT CloudHSM keys.
- **EC2 Dedicated Instances**: run in a VPC on hardware that's dedicated to a single customer. Dedicated instances may still share hardware with other non-dedicated instances from the same AWS account.
- **EC2 Dedicated Hosts**: same as above AND provides additional visibility and control over how instances are placed on a physical server + consistent deployment to same physical server each time + enables you to use existing server-bound licenses (e.g VMWare, Oracle which may require dedicated hosts) + allows you to address corporate and regulatory compliance.

AWS EC2 Hypervisor: is software, firmware, hardware that creates and runs virtual machines.

- **Hypervisor access by AWS employees**: Admins require MFA, access is logged/audited, administration hosts are specifically designed/built/configured/hardened to protect the management plane. Access is revoked upon no more business need for employee.

- **Memory scrubbing**:
    - EBS volumes are NOT scrubbed immediately after being deleted, only PRIOR TO BEING RE-USED.
    - Host/guest (EC2) memory that is allocated is scrubbed/zeroed by the Hypervisor as soon as it is UNALLOCATED from the guest. The memory is not returned to the pool of free memory until scrubbing is complete.
- **AWS EC2 Hypervisor**: is software, firmware, hardware that creates and runs virtual machines. EC2 AMIs run on 2 types of virtualisation:
    - **Hardware Virtual Machine (HVM)**: VM guests are fully virtualised - they are not aware that they're sharing processing time with other VMs.
    - **Paravirtual (PV)**: (MORE LIGHTWEIGHT / QUICKER) VM guests relies on hypervisor to provide support for operations that normally require privileged access = guest OS has no elevated CPU access.
    - **Hypervisor access by AWS employees** is logged/audited + requires MFA + access strictly controlled. This cloud management plane is specially designed, built, configured and hardened.
    - **Guest OS (EC2) are controlled by customers** with full root over accounts, services and apps running on EC2. AWS has no right to access EC2s.
    - *AWS IS NOW SHIFTING ITS PHYSICAL SERVERS FROM XEN HYPERVISOR TO LINUX KERNEL-BASED VIRTUAL MACHINE (KVM) OPEN-SOURCE HYPERVISOR.*

EC2 - Auto Scaling

- **EC2 Auto Scaling** helps ensure you have the correct min/max number of EC2 instances available to handle load for your application. Collections of EC2 instances = **Auto Scaling Groups**.
- **VPC Endpoint for Auto Scaling** is available to allow you to call the EC2 Auto Scaling API from within your VPC without sending traffic over the internet (powered by **AWS PrivateLink**).
- **Security Groups** apply to an Auto Scaling group = all instances within ASG are subject to the Security Group rules.

EC2 - Internal instance logging

- **Record running processes**: by using AWS Systems Manager Run Command to send a list of running processes to an S3 bucket.
- *CloudWatch CANNOT record running processes in an EC2, only EC2 metrics like CPU utilization.*

Container security:

1. **Don't store secrets**
   - Use IAM roles instead of hardcoding user credentials.
   - Use Secrets Manager for RDS credentials and API keys.
   - Use Amazon Certificate Manager (ACM) if you have TLS certs to store and manage.
2. **Don't run container as AWS root**
   - Don't run containers using your AWS Root account.
3. **Less is more**
   - Minimise attack surface by running one service per container, not multiple per container.
   - Avoid unnecessary libraries: remove code/libraries you don't need in your container image.
4. **Use trusted images only**
   - Avoid public repos, where you don't know the origin of code.
   - Use images from a trusted source or ones created in-house.
   - Scan for CVEs using Amazon Inspector or external tools.
   - **AWS Elastic Container Registry (ECR)** to store your own container images and use **AWS Elastic Container Service (ECS)** to run containers.
5. **Infrastructure security**
   - Avoid public internet by using **ECS Interface Endpoints** (similar to VPC endpoints)
   - If using public internet, use TLS to secure end-to-end communication between end-users and your apps running in containers.
   - **Amazon Certificate Manager (ACM)** can be used to provide single, central interface for storing and managing certificates. It integrates well with many AWS services.

Amazon DNS - Default DHCP vs. Custom DHCP options

- **Default DHCP** options set uses `AmazonProvidedDNS`. You cannot update the existing option set, only delete and create a new one.
- **Custom DHCP** options set can be used by creating a NEW SET of DHCP options and providing IPs of up to 4 DNS servers.

Chapter 5 - Data Protection With VPCs

AWS Virtual Private Cloud (VPC) lets you provision a logically isolated section of the AWS cloud where you can launch resources in a virtual network that you define.

- Flow of inbound traffic: entry via. **VPC Virtual Private Gateway (VPN)** or **VPC Internet Gateway (public)** -> Route Tables -> Network ACL -> Subnet -> Security Groups -> EC2s.
- **VPC Peering** allows you to connect one VPC with another VPC via. direct network route using private IP addresses.
  - Peering is in a STAR CONFIGURATION i.e. 1 central VPC with 4 others. No TRANSITIVE PEERING is allowed.
- Setting up and testing a custom VPC:
  i. Create VPC -> provision private/public subnets to VPC -> provision Internet Gateway for public internet connectivity to VPC
  ii. Create CUSTOM ROUTE TABLE -> add route to the internet `0.0.0.0/0` via. Internet Gateway -> disable internet access for MAIN ROUTE TABLE, so all new subnets created won't have internet access by default -> associate subnet with CUSTOM ROUTE TABLE
  iii. Test internet connectivity using EC2s: Turn on `Auto-assign public IP addresses` for the public subnet so a public IPv4 address is assigned for all EC2s launched into the subnet -> try to SSH into EC2 in public subnet.

VPC - Route Table basics

- **RT Destination** is the IP range (CIDR) of addresses that you want traffic to end up.
- **RT Target** is the Gateway, Network Interface or connection to send the traffic through towards the RT Destination.
- **Local Route** is the default route for communication within the VPC.
- Example Route Table Routes:
  - `Dest: 10.0.0.0/16 | Tar: Local`: Local route within the VPC CIDR.
  - `Dest: 172.31.0.0/16 | Tar: pcx-123`: Route to a secondary VPC (CIDR `172.31.0.0/16`), via. a VPC Peering Connection.
  - `Dest: 0.0.0.0/0 | Tar: igw-321`: Route for IPv4 traffic to the public internet, via. Internet Gateway.
  - `Dest: ::/0 | Tar: eigw-456`: Route for IPv6 traffic to the public internet, via. an Egress-only Internet Gateway (outbound-only communications).

VPC - AWS NAT Instance / AWS NAT Gateway

- A NAT device forwards traffic from instances in the private subnet to the internet / AWS services, then sends the response back to the instances. The internet cannot initiate connections with these instances.
- When traffic goes to the internet, source IPv4 address is replaced with the NAT device address and response traffic is translated by NAT device back to the instance's private IPv4 addresses.

VPC - AWS NAT Instances (OLD NAT METHOD)

- **Traffic flow**: EC2s in private subnet -> route table -> NAT instance in public subnet -> Internet Gateway -> the internet.
- **Single instance reliance**: any crash = no internet access for servers in private subnet.
- **Limited network throughput**: amount of traffic supported depends on instance size.
- To have high availability, requires using Autoscaling Groups + multiple subnets in different AZs + scripts to automate failover (switching to a standby server upon failure).
- NAT instances can be used as a **Bastion Server**.

VPC - AWS NAT Gateways (PREFERRED NAT METHOD)

- **Traffic flow**: EC2s in private subnet -> route table -> NAT Gateway in public subnet -> Internet Gateway -> the internet.
- **Security is managed by AWS**: no need for Security Groups, server patching, antivirus protections etc.
- Automatically assigned with a public IP. Scales automatically to 10GBps. Highly available, automatic failover.
- Create at least 1 NAT Gateway per Availability Zone so there is redundancy in case of Zone Failure.
- GuardDuty can monitor NAT Gateway metrics.

VPC - Internet Gateway

- **Internet Gateway** is a VPC component that allows communication between your VPC and the Internet.
- **Egress only Internet Gateway** is a VPC component that allows outbound communication over IPv6 from instances in your VPC to the internet, but prevents the internet from initiating an IPv6 connection with your instances.

VPC Flow Logs enable you to capture info about IP traffic (+ traffic metadata) going to/from a VPC, VPC's subnet or an ENI.

- **Flow log storage** is in CloudWatch Logs log groups.
- **Flow log creation** are at 3 different levels: (1) VPC - captures all ENI traffic (2) Subnet - capture ENI and EC2 traffic within a particular subnet (3) Network Interface
- **Limitations**: Flow Logs for peered VPCs must be in the same AWS account. Flow logs can't be reconfigured after creation. Not all traffic is monitored (e.g. EC2 metadata, DHCP traffic, traffic to AWS DNS server, traffic to reserved AWS IPs)

VPC Endpoints enable you to privately connect (using **AWS PrivateLink**) your VPC to supported AWS services without needing a NAT Gateway (goes over private network).

- **MOST SECURE WAY TO ALLOW RESOURCES TO CONNECT TO OTHER AWS SERVICES, AS TRAFFIC NEVER LEAVES VPC**.
- **VPC Endpoint policies** are used to control access to the AWS service which you are connecting to.
- EC2 instances in your VPC do NOT require public IPs to communicate with resources in supported VPC Endpoint services.
- Supported services include `S3, DynamoDB, SNS, ELBs, CloudFormation` and more.
- Restrict access to AWS resources to only specific VPC Endpoints by using `aws:sourceVpce: vpce-endpoint-id`. E.g. S3 Bucket Policy condition to access S3 bucket via. specific VPCE only.s

NACLs vs. Security Groups

- **NACLs are STATELESS**: responses to allowed inbound traffic are subject to outbound rules (vice versa).
- **SGs are STATEFUL**: response to outbound requests will be allowed to flow in regardless of inbound security group rules (vice versa).
- **Default VPC NACL** will ALLOW ALL traffic in/out of subnets associated with the VPC.
- **Custom NACLs** created will by default DENY ALL traffic in/out.
- You can block specific IP addresses using NACLs, but not with Security Groups.

Elastic Load Balancers and TLS/SSL Termination: terminate at ELB vs. EC2

- **Terminate at Load Balancer**: ALB decrypts HTTPS request -> inspects headers -> routes request to EC2 as plaintext over local private network in your VPC.

    - **PRO: More resources + more cost-effective (USE ALB)** as decryption is offloaded to ALB, allowing you to use smaller EC2 instances to handle application load.
    - **PRO: Reduces administrative overhead (USE ALB)** as you don't need to manage X509 certs (used to decrypt/encrypt) individually on each EC2.
    - **CON: Unencrypted traffic (USE NLB OR CLB)** between ALB and EC2 (however AWS states that EC2s that aren't part of the connection cannot listen in, even if they are running within your AWS account).
    - **CON: Compliance/regulatory requirements (USE NLB OR CLB)** typically require use of E2E encryption.
- **Application Load Balancer (HTTP/HTTPS)** only supports HTTPS termination using an SSL cert on the Load Balancer itself. Only supports HTTP/HTTPS connections.
- **Network Load Balancer (TCP, UDP, TLS)** supports TLS/SSL termination on the Load Balancer AND EC2 instances. You will need to use TCP (load balancing at TCP transport-layer rather than HTTP application-layer).
- **Classic Load Balancer (TCP, SSL/TLS, HTTP, HTTPS)** supports TLS/SSL termination on the Load Balancer AND EC2 instances.

How to build a highly available Bastion instance:

- **High availability**: at least 2x Bastion instances in 2 public subnets in 2 Availability Zones.
- **Autoscaling Groups**: minimum of 1 Bastion -> if Bastion goes down, ASG deploys a Bastion into one AZ or another.
- **Route53 health check**: run health checks on Bastion server.

AWS Systems Manager - Session Manager: enables secure remote login to EC2 instances (alternative to RDP/SSH)

- **Session Manager is Secure**: TLS encryption, no Bastion required, no Security Groups needed, CloudTrail logging, keystroke logging sent to CloudWatch/S3.
- Setting up Session Manager in AWS Systems Manager:
    i. Create EC2 instance role w/ permission to call Systems Manager + install SSM Agent on EC2.
    ii. Create CloudWatch Log Group `SM_LogGroup` + associate CW Log Group with Session Manager.
    iii. Configure Session Manager logging: encrypt session logs with KMS CMK (user of session + EC2 instance role must have access to CMK)
    iv. Configure Session Manager logging: choose to send logs to the CloudWatch Log Group OR an S3 bucket.

v. Start a session inside Session Manager Console -> launch web shell.

AWS Systems Manager - Patch Manager automates the process of patching managed instances

- Use Patch Manager to generate a report of out-of-compliance (unpatched) instances/servers.
- Use Patch Manager to install missing patches.

AWS CloudHSM provides Hardware Security Modules (HSM) in a cluster - a collection of individual HSMs that AWS CloudHSM keeps in sync. Any tasks performed on one HSM, other HSMs in the cluster will be updated.

- CloudHSM user types:
  i. **Precrypto Officer (PRECO)**: default account with admin/pass creds -> upon setting password, you will be promoted to CO.
  ii. **Crypto Officer (CO)**: performs user management e.g. create and delete users and change user passwords.
  iii. **Crypto Users (CU)**: performs key management (*create/delete/import/export*) and cryptographic operations (*encr/decr/sign/verify*).
  iv. **Appliance User (AU)**: performs cloning and synchronization operations. CloudHSM uses AU to sync HSMs. AU exists in all HSMs and has limited permissions.

AWS Direct Connect (DX) enables you to establish a dedicated private connection from an *on-premise network/datacenter to 1+ VPCs in the same region* to reduce network costs, increase throughput, provide more consistent network experience than internet-based connections.

- **DX + VPN (AWS Virtual Private Gateway endpoint)**: The VPN provides end-to-end encryption while AWS Direct Connect provides a reliable network with low latency and increased bandwidth.

AWS Transit Gateway connects VPCs and on-premise datacenters/networks through a central hub. Acts as a cloud router.

- NOT using Transit Gateway
  - Each VPC requires VPN connection and config to the on-prem network.
  - VPCs require peering between each other. If hundreds of VPCs = difficult to manage, not scalable.
- USING Transit Gateway

- **Highly scalable**: supports thousands of VPCs (hub-and-spoke architecture)
- **Centralised**: Transit Gateway sits between all your VPCs and Datacentre, only needs to be configured once. Any VPC connected to Transit Gateway can communicate with every other connected VPC.
- **Route Tables**: can be used to enforce which VPCs can communicate with each other.
- **Secure**: communication between VPCs are done via. AWS private network. Inter-region traffic is supported.

Chapter 6 - Incident Response and AWS in the Real World

DDoS: Amplification / Reflection Attacks: Attacker sends 3rd-party server a request using spoofed IP -> server responds with greater payload than inital request.

Minimising DDoS

1. **Minimise attack surface**: reduce internet accessible services/servers, use Bastion host, whitelist allowed IPs.
2. **Absorb attack by scaling**: scale horizontally (machines++) and vertically (compute++) = additional levels of redundancy and buys time to analyze the attack.
3. **Safeguard public-facing resources**: AWS WAF, CloudFront (Geo-blocking, S3 Origin Access Identity), Route53 (alias records to redirect traffic to CloudFront, ELB or other security tools + Private DNS to manage internal DNS names for DBs, webservers etc. without exposing info publically).
4. **Learn what normal behaviour looks like**: spot abnormalities, create alarms to alert for unusual behaviour, collect forensic data to understand attacks.
5. **Create a plan for attacks**: validate design of architecture, understand costs of resiliency, know who to contact when attack occurs.
6. **AWS Shield**: protects all AWS customers on ELB, CloudFront and Route53 against SYN/UDP floods, reflection attacks and other layer 3/4 DDoS attacks.
7. **AWS Shield Advanced**: enhanced protections, $3k/month, always-on flow-based monitoring of network/app traffic, 24/7 DDoS Response Team (DRT), AWS billing protection.

AWS Account compromised - what to do?

1. CHANGE `AWS account Root Password.`
2. ROTATE, then DELETE `Root and IAM user access keys.`

3. DELETE `IAM Users that have been potentially compromised.`
4. DELETE `AWS resources you didn't create.`

EC2 has been hacked - what to do?

1. Stop instances immediately.
2. Take a snapshot of EBS volume + terminate instance.
3. Deploy a copy of the instance in an **isolated environment** (isolated VPC, no internet access).
4. Access the instance using an **isolated forensic workstation**.
5. Read logs to figure out how they obtained access.

Leaked Github keys - what to do?

- **IAM User Credentials**: (1) De-activate IAM User Access Key (2) Create new User Access Key (3) Delete old User Access Key
- **Root User Credentials**: (1) Goto `My Security Credentials -> Access Keys ->` De-active Root User Access Key. (2) Create new Root User Access Key (3) Delete old Root User Access Key

AWS Certificate Manager (ACM): provision a SSL cert for a domain name you have registered. SSL certs are autorenewed provided the domain name was purchased from Route53.

- **Requesting a SSL cert**: (1) Add domain name (2) Select domain validation methods DNS or EMAIL (3) If DNS validation, add the given CNAME record to Route53. *IF YOU REQUIRE HTTPS BETWEEN END-USERS <-> CF, CERTIFICATE CAN ONLY BE REQUESTED/IMPORTED ON US-EAST-1 IN ACM*
- **Auto-renew SSL/TLS certs**: ACM provides autorenewal for Amazon-issued SSL/TLS certs.
- **Manual-renew SSL/TLS certs**: Imported SSL/TLS certs OR certs associated with R53 private hosted zones must be manually renewed.
- **Use Amazon SSL cert with CloudFront**: Goto `CloudFront` -> select distribution -> edit settings to change default CloudFront SSL cert to the new custom SSL cert associated with your domain name.
- **Use Amazon SSL cert with EC2**: Goto `EC2` -> `Load Balancers` -> create a load balancer -> `choose a certificate from ACM`.

Configuring Security Policy (SSL/TLS protocols and ciphers) with ELBs / CloudFront

- **2016-08** is the recommended Security Policy as it supports most ciphers.
- **ECDHE-* cipher** is required to enable Perfect Forward Secrecy.

- **Perfect Forward Secrecy** is a concept where PAST captured-data cannot be decrypted using a compromised private key, as a new key is created for each SSL-session. The compromised key would only be able to decode data for its specific session, but no other.

API Gateway - Throttling and Caching

- **Steady-State Limit**: 10,000 req/sec
- **Burst Limit**: (max concurrent requests) 5,000 req across all APIs within an AWS account.
- **API Gateway Caching**: cache API endpoint response for a specified **TIME TO LIVE (TTL)**
    - TTL=300 (default) | TTL=3600 (max) | TTL=0 (cache disabled)

AWS Systems Manager - Run Command

- **Manage EC2s and on-premise systems**: automate admin tasks and adhoc config changes e.g. patching.
- **Using EC2 Run Command**: Create EC2 instance and role for SSM `EC2 role for Simple Systems Manager` -> Goto `SSM` -> `Run a command` -> choose a command document e.g. `Configure CloudWatch` -> select target instance and run.
- **SSM Agent** needs to be installed AND **EC2 instance role with SSM permissions** enabled for the Run Command to work.
- **Systems Manager Document** defines the commands and parameters to be run.

Compliance Frameworks

- **ISO27001**: *establishing, implementing, operating, monitoring, reviewing, maintaining and improving documented Information Security Management System (ISMS)* within the context of the organisation's overall business risks.
- **FedRAMP (Federal Risk and Authorization Management Platform)**: Government-wide program that provides a standardised approach to security assessment, authorisation, continuous monitoring for cloud products/services.
- **HIPAA (Federal Health Insurance Portability and Accountability Act of 1996)**: *lower cost of healthcare and ensure good data security around healthcare info.*
- **NIST (National Institute of Standards and Technology)**: A framework for improving critical infrastructure security for organisations.
- **PCI DSS (Payment Card Industry Data Security Standard)**: Policies and procedures to optimise security of credit/debit/cash card transactions and protect cardholders against misuse of personal info.

- **FIPS 140-2**: a U.S government computer security standard used to approve cryptograhic modules.
  - **AWS CloudHSM** meets level 3. Rated from level 1 -> level 4 (highest).

Chapter 7 - Additional Topics

Using Amazon Macie

- Macie can only monitor S3 buckets within the same region.
- Macie uses `AWSServiceRoleForAmazonMacie` which cover mainly permissions for CloudTrail (creating/reading logs) and S3 (creating/deleting buckets and objects)
- An S3 CloudTrail bucket will be created to capture all data events with Macie.

Using Amazon GuardDuty

- Takes 7-14 days to establish a baseline - "*what is normal behaviour in your account?*"
- 30 days free, then is charged off **quantity of CloudTrail events** and **volume of DNS and VPC Flow Logs**.

AWS Secrets Manager

- **Store credentials** for RDS, non-RDS databases (DynamoDB) and any other secrets as long as you can store them as a key-value pair (SSH keys, API keys).
- **Automatic secret rotation** can be turned on, but make sure your app is not using hardcoded credentials + make sure it is retrieving credentials from Secrets Manager.
- **Deletion of secrets** require a 7 day waiting period.
- **Secrets Manager BENEFITS vs. Parameter Store**:
  - **Generation of passwords**: Secrets Manager can generate random passwords.
  - **Secrets rotation**: Secrets Manager can natively rotate RDS passwords.
  - **Cross-account access**: Secrets Manager secrets can be accessed cross-account.
- **Parameter Store** is for storing:
  - Plaintext (`String`, `StringList`) data such as non-confidential environment vars or config values.
  - Encrypted (`SecureString`) data such as passwords, license codes, application secrets (encrypted with AWS or Customer managed CMKs).

Using AWS Simple Email Service (SES)

- Configure Security Group associated with EC2 to allow outbound to the SES SMTP endpoint.
- **Using SES**: if you have reached a Sending Limit, you can request to increase it via. AWS Support.
- **Using custom Mail Transmission software**: e.g. JavaMail. EC2 throttles traffic over the default SMTP `port 25`. You can bypass the throttle by using `port 587` or `port 2587`.

AWS Security Hub:

1. **Centralised dashboard** for findings/alerts from key AWS security services.
2. **Automated compliance checks** by evaluating AWS resources against PCI-DSS, CIS controls and AWS Foundational Security Best Practices.

- Integrates with *GuardDuty, Macie, Inspector, IAM Access Analyzer, Firewall Manager, 3rd-party marketplace tools, CloudWatch (trigger lambdas/SIEM/3rd-party tools)*.

Network packet inspection in AWS

- **NO AWS SERVICE SUPPORTS NETWORK PACKET INSPECTION - USE 3RD-PARTY SOLUTION FROM AWS MARKETPLACE**.
- **Network Packet Inspection / Deep Packet Inspection** involves inspecting a packet's headers and data.
  - Filters non-compliant protocols, viruses, spam, intrusions.
  - Takes action by blocking, re-routing or logging.
  - IDS/IPS combined with a traditional firewall.
- How to use: install 3rd-party solution for Network Packet Inspection via. AWS Marketplace.

Active Directory Federation with AWS: AWS enables federated sign-in to AWS using Active Directory credentials

- Great for companies with an existing Active Directory Domain + have corporate users who have AD accounts.
- **2-WAY TRUST**: establishing AD federation with AWS
  - In AWS, configure ADFS as the **Trusted Identity Provider** = "*Trust ADFS to provide your users' identities*"
  - In ADFS, configure AWS as the **Trusted Relying Party** = "*Trust AWS to consume your users' identities*"
- Using ADFS to sign-in to AWS Console:
  i. User logs into ADFS via. ADFS sign-in page + provide credentials.

ii. ADFS authenticates user against Active Directory.
iii. ADFS sends back authentication response to user in the form of a SAML token.
iv. User sends SAML token to AWS sign-in endpoint (choose / assume role page).
v. AWS sign-in endpoint makes an `STS AssumeRoleWithSAML` request to get temporary creds to AWS -> STS returns temporary credentials.
vi. AWS sign-in endpoint redirects user to the AWS Console.

AWS Artifact: is a central resource for compliance and security related documents / information

- Demonstrate compliance to regulators, evaluate your own cloud architecture, assess effectiveness of internal controls.
- Download *ISO 1270001 certs, PCI-DSS docs, SOC reports*.

AWS Kinesis - security

- **Kinesis Data Streams SSE** automatically encrypts data BEFORE its at rest using an AWS CMK you specify. The producers/consumers of your Kinesis stream don't need to manage the keys or perform cryptographic operations.
- **VPC Endpoints** can be used with your Kinesis streams so traffic won't leave the Amazon network.
- **API calls/requests** must use min `TLS 1.0`, use cipher suites with Perfect Forward Secrecy such as `ECDHE`. Requests must also be signed using an `AccessKeyId/SecretAccessKey` or `STS temporary security credentials (sts:AssumeRole)`.

Troubleshooting Scenarios

VPC Peering - connection issues between VPCs (looks at Route Table, NACL/SG rules)

1. Verify that routes in **Routing Tables** for ALL peered VPCs and configured correctly, so they know how to route traffic to each other.
2. Verify that an ALLOW rule exists in the **NACL table** for the required traffic.
3. Verify that **Security Group rules** allow traffic between the peered VPCs.
4. Verify using **VPC Flow Logs**.

VPC - no internet access

- Configure Routing Table to forward traffic / use an **Internet Gateway** or a **NAT Gateway**.

VPC - VPN connection not working

- Ensure Routing Table is routing traffic to your data center via. the **Virtual Private Gateway**.
- VPN traffic flow: VPC Virtual Private Gateway -> VPC Router -> Routing Table -> NACL -> Private Subnet -> SG -> EC2

CloudWatch Logs - Lambda / EC2 not logging to CloudWatch Logs

- Basic role permissions required to log to CloudWatch are: `CreateLogGroup`, `CreateLogStream` and `PutLogEvents`.
- EC2 requires a CloudWatch agent installed and running.
- Lambda's EXECUTION ROLE requires the above permissions.

CloudWatch Events / S3 Event - Event not invoking Lambda

- Add permissions in **Lambda Function Policy** for Cloudwatch Events or S3 Events to `invoke` your Lambda function.
- This applies to ANY services that can invoke Lambdas listed here: https://aws.amazon.com/blogs/architecture/understanding-the-different-ways-to-invoke-lambda-functions

CloudTrail Logging issues

- **CT Logging not working**: Check S3 bucket name, S3 Bucket Policy, S3 Access Control List.
- **CT Logging is expensive**: S3 Data Events record all object-level API activity. Lambda Data Events record all invoke-API operations.
- **Auditor can't access logs**: `AWSCloudTrailReadOnlyAccess` IAM Policy allows access to CloudTrail logs.

Troubleshooting Identity Federation - use the correct API for the job

- `sts:AssumeRole`: If authenticated by AWS, typically for cross-account delegation.
- `sts:AssumeRoleWithSAML`: If authenticated by a SAML IdP (Active Directory etc.).
- `sts:AssumeRoleWithWebIdentity`: If authenticated by a Web Identity Provider (Facebook,, Google etc.)

SMTP or AWS SES SMTP - timeout issues

- EC2 throttles SMTP `port 25` by default. Use `port 587` or `port 2587` which are unrestricted or request to remove email sending limitations.

Lambda - issues with Lambda not being invoked OR Lambda not accessing resource

- **Lambda executions not logged in CloudWatch Logs**: `CreateLogGroup`, `CreateLogStream` and `PutLogEvents` required in LAMBDA EXECUTION ROLE.
- **Lambda cannot perform an action** e.g. write to S3, log to CloudWatch, Terminate Instances, use a CMK, use Secrets Manager
  - Check the LAMBDA EXECUTION ROLE allows the actions.
- **Lambda cannot be invoked by service** e.g. CloudWatch Event invoking Lambda.
  - Check the LAMBDA FUNCTION POLICY allows the service.
- Remember that some services have their own resource-based policies which will impact access to the resource
  - E.g S3 Bucket Policy, KMS Key Policies etc.
- NOTE: *Lambda Execution Role* defines what the Lambda fn can do.
- NOTE: *Function Policy* defines which services can invoke the Lambda fn.

AWS Systems Manager Run Command - Run Command not executing on some instances

1. Ensure the Security Groups allow outbound communication for the instances - SSM agents in the EC2s require communication to the SSM endpoint.
2. Check the `/var/log/amazon/ssm/errors.log` file - the file shows if there are SSM agent errors.
3. Ensure the SSM agent is running on the target machine and verify you are running a supported machine type - SSM agent has to be running for the Run Command to work on the EC2.

S3 pre-signed URLs - pre-signed URLs expiring too fast / before specified expiry time?

- If pre-signed URL was generated by an IAM Role / STS temporary security credentials that have shorter session expiry than pre-signed URL expiry, then it will override the pre-signed URL expiry.
- Use an IAM user with long-term credentials if you want longer expiry for S3 pre-signed URLs.

CloudFront signed-URLS vs. signed-Cookies: allow you to control who can access your content

- **CF Signed-URLS**: for restricting access to INDIVIDUAL FILES (e.g. `.exe` install file) of if users are on a custom-HTTP client that does not support cookies.

- **CF Signed-Cookies**: for restricting access to MULTIPLE RESTRICTED FILES (e.g. access control on objects specific to a user's account).

AWS Best Practices for DDoS Resiliency

Original link to resource: https://d1.awsstatic.com/whitepapers/Security/DDoS_White_Paper.pdf

DDoS Attacks

Distributed Denial of Service (DDoS) is where an attacker uses multiple sources, such as distributed groups of malware infected computers, routers, IoT devices and other endpoints to orchestrate an attack against a target, preventing legitimate users from accessing the target.

Infrastructure Layer Attacks (OSI3 - Network / OSI4 - Transport)

- **UDP Reflection**
  - (1) Craft UDP packet with target IP = source IP (2) Send malicious UDP packet to intermediate server (3) intermediate server is tricked into sending UDP response packets to target.
  - Amplification factor: 64byte request to 128byte response = x2 amplification
- **SYN Flood**
  - Exploits 3-way handshake
  - (1) Send flood of SYN packets to target, but never final ACK (2) Target waits waits for response to half-open TCP

Application Layer Attacks (OSI7 - Application)

- **HTTP Flood**
  - Send HTTP requests targeted at specific resource or emulating human interactions.
- **Cache-Busting**
  - Force CDN to bypass cache and to retrieve data from origin server for every request, causing strain on appserver.
- **WordPress XML-RPC Flood (WordPress pingback)**
  - Exploit XML-RPC API of a WordPress site.

    - (1) Attacker_WP notifies Target_WP of a site link via. pingback feature
      (2) Target_WP attempts to fetch Attacker_WP to verify existence of link
      (3) Target_WP is flooded.
- Other attacks that can impact availability
    - Scraper bots, brute-forcing, credential-stuffing.

DDoS Mitigation Techniques

AWS Shield (Standard)

- Provided by default to AWS customers, on all AWS services in every AWS region.
- Defends against common network and transport layer DDoS attacks.

AWS Shield (Advanced)

- Access to **AWS DDoS Response Team** (DRT) for assistance in mitigating DDoS attacks that impact application availability.
- Access to **Global Threat Environment** dashboard, providing an overview of DDoS attacks observed and mitigated by AWS.
- Access to **AWS WAF** at NO ADDITIONAL COST for mitigating application-layer DDoS (when used with CloudFront or ALB).
- Access to **AWS Firewall Manager** at NO ADDITIONAL COST for automated policy enforcement.
- **Sensitive detection thresholds** which routes traffic into DDoS mitigation system earlier and can **improve time-to-mitigate attacks** against AWS EC2, LNB.
- **Cost Protection** that allows you to request a limited refund of scaling-related costs that result from DDoS.
- **Enhanced service level agreement**.

Infrastructure Layer Defenses

- **EC2 Auto Scaling**
    - Sudden traffic surge -> CloudWatch alarm initiates Auto Scaling based on CPU / RAM / NetworkIO / custom metrics -> EC2 fleet size increase (increase in number of EC2s)
- **Choice of Region**
    - Choose regions that are close to internet exchanges where international carriers have a strong presence, to help give you internet capacity to mitigate much larger DDoS attacks.

- **Elastic Load Balancing**
    - Distribute traffic across many instances to reduce excess traffic.
    - ELB scales automatically: attach ELB to an Auto Scaling Group.
    - 3 types of ELB: (1) Application (web-apps) (2) Classic (3) Network (TCP-based apps).
- Leverage **AWS Edge Locations** for Scale
    - When a user requests content that you're serving with CloudFront, they are routed to the EDGE LOCATION that provides the lowest latency. EDGE LOCATIONS are a worldwide network of data centers.
    - **Web Application Delivery at the Edge**:
        - Reduce number of TCP connections to your origin (preventing HTTP Floods).
        - Prevent SYN Floods and UDP Reflection attacks from reaching your origin as CloudFront only accepts well-formed connections.
        - When serving static content with S3, use CloudFront to protect your bucket via. **Origin Access Identity (OAI)** to ensure users can only access S3 objects by using CloudFront URLs.
    - **Domain Name Resolution at the Edge**:
        - Route53 has features such as Traffic Flow, Latency Based Routing, Geo DNS, Health Checks and Monitoring to allow you to control how R53 responds to DNS requests, to improve app performance and prevent outages.
        - Detect anomalies in the source and volume of DNS queries and prioritize requests from users that are known to be reliable.

Application Layer Defenses

- Detect and Filter Malicious Web Requests
    - Use **AWS CloudFront** to (1) cache static content and serve it from AWS Edge Locations (2) prevent non-web traffic from reaching your origin to reduce server load (3) automatically close connections from slow read/write attackers
    - Use **AWS WAF** to filter and block requests based on IP match, rate-based, regex rules defined by yourself, managed by AWS or 3rd-party marketplace rules.
    - Use **AWS Shield Advanced** to engage AWS DDoS Response Team (DRT) to create rules to mitigate an attack that is impacting your application.
    - Use **AWS Firewall Manager** to centrally configure and manage WAF rules across your organisation. Your AWS Organisations master account

can designate an administrator account, which is authorized to create Firewall Manager policies.

Attack Surface Reduction

Resources that are NOT EXPOSED TO THE INTERNET are more difficult to attack, limiting the options an attacker has to target your application's availability.

Blocking access to origin #1: by using **Security Groups** and **Network Access Control Lists**.

- Security Group Example: Webapp that uses an ELB and several EC2s
    - i. Create an SG for the ELB and SG for the instances.
    - ii. Create an ALLOW rule to permit internet traffic to ELB SG + rule to permit traffic from ELB SG to the EC2s' SG = more difficult for attacker to learn about and impact the webapp.
- NACL: Use NACLs to explicitly deny certain types of traffic e.g. deny certain CIDR ranges, protocols, signatures based off known DDoS IPs etc.
- Use **AWS Shield Advanced to register Elastic IPs as Protected Resources**. DDoS against EIPs will be detected more quickly, resulting in faster mitigation.

Blocking access to origin #2: by **only allowing requests from CloudFront**. Malicious traffic cannot bypass CF and WAF and hit your origin directly.

1. Create a Security Group, allowing only traffic from CloudFront to your ELB or EC2s.
2. Create Lambda to update SG rules dynamically, triggered by an `AmazonIPSpaceChanged` SNS topic (AWS updating their internal IP ranges).
3. Use the `X-Shared-Secret` header to validate that requests sent to your origin are coming from CloudFront.

Protecting API endpoints: by using **Amazon API Gateway**.

- Configure CF distributions to include the a custom header `x-api-key`, sent to your origin endpoint.
- Configure standard or burst rate limits for each REST API method.

Operational Techniques

It is useful to know WHEN a DDoS attack is targeting your application so you can take mitigation steps.

Visibility

- **Amazon CloudWatch** to monitor apps running on AWS - collect and track metrics, log files, set alarms and automatically respond to changes in your AWS resources. E.g. AWS WAF `BlockedRequests` or `CountedRequests` metrics.
- **AWS Shield Advanced** provides additional metrics to indicate if your app is being targeted. E.g. `DDoSDetected` or volume-based metrics `DDoSAttackBitsPerSecond`, `DDoSAttackPacketsPerSecond` or `DDoSAttackRequestsPerSecond`. Can be integrated with CloudWatch or 3rd-party tools e.g. Slack/PagerDuty.
- **VPC Flow Logs** to capture information about the IP traffic going to and from your network interfaces in your VPC.
    - Each flow records *src ip, dest ip, src port, dest port, protocol, no. of packets and bytes transferred*. Use this info to identify anomalies in network traffic and to identify a specific attack vector. E.g. UDP reflection attacks = src port 53 for DNS reflection.

Support

- Subscribe to **Business Support** to get 24x7 access to Cloud Support Engineers who can assist with DDoS attack issues.
- Subscribe to **Enterprise Support** for the ability to open CRITICAL CASES and receive the FASTEST RESPONSE from a Senior Cloud Support Engineer.
- Subscribe to **AWS Shield Advanced** to escalate cases to the **AWS DDoS Response Team (DRT)**.
- Use the **AWS Shield Engagement Lambda** to more quickly initiate contact with the DRT. E.g. use an AWS IoT button to trigger the AWS Lambda function if you have an emergency situation (emergency panic red button).

AWS Key Management Service Best Practices

Identity and Access Management

AWS KMS and IAM policies - use IAM Policies in combination with Key Policies to control access to CMKs

- **identity-based policy**: policy attached to IAM entities (users, groups, roles).
- **resource-based policy**: policy attached to resources OUTSIDE of IAM.

- IAM policies are not enough by themselves to allow access to a CMK, but can be used *IN COMBINATION with a Key Policy* to grant access. To do this, ensure that the CMK Key Policy includes a *POLICY STATEMENT that enables IAM policies*.

Key Policy - a resource-based policy attached to CMKs which control access to the CMK

- All CMKs have a Key Policy.
- To access an encrypted resource: (1) Principal needs permissions to use the resource (2) Principal needs permission to use the encryption key that protects the resource
- `kms:viaService`: constrain CMK access so that it can only be used specified AWS services.

Key Policy Example - create and delegate use of an encrypted Amazon Elastic Block Store (EBS) volume to an EC2.

- **CMK Grants** are used to delegate subset of permissions to AWS services/principals to use your keys.

```
// Allow IAM principal to generate a data key (encrypted by CMK) + decrypt data
key (using same CMK)
// Data key: used to encrypt data.
{
    "Sid": "Allow for use of this Key", // sid = a description for policy
statements
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/UserRole"
    },
    "Action": [
        "kms:GenerateDataKeyWithoutPlaintext",  // returns a unique symmetric data
key (encrypted by CMK)
        "kms:Decrypt"
    ],
    "Resource": "*"
},
// Allow IAM principal to create, list, revoke CMK Grants for EC2 service.
// EC2 will use delegated permissions to access an encrypted EBS volume, to re-
attach it back to an instance if the volume gets detached due to a planned or
unplanned outage.
{
    "Sid": "Allow for EC2 Use",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/UserRole"
    },
    "Action": [
        "kms:CreateGrant", // adds grant to CMK, allowing a GRANTEE principal to
use the CMK when conditions of grant are met.
        "kms:ListGrants",
        "kms:RevokeGrant"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:ViaService": "ec2.us-west-2.amazonaws.com" // only EC2 can use
the created grants
        }
    }
}
```

Key Policies - Least Privilege / Separation of Duties

- **Ensure Separation of Duty by NOT using "kms:*"**: in an IAM or Key Policy: this grants both ADMINISTRATIVE and USAGE permissions on all CMKs to which the principal has access to. Users with `kms:PutKeyPolicy` permission for a CMK can completely replace the Key Policy.
- **Ensure "Effect":"Deny" is NOT used with "NotPrincipal"**: permissions are explicitly denied to all principals EXCEPT for the principals specified under `NotPrincipal`.

Cross Account Sharing of Keys (2 steps)

1. **Key Policy** for the CMK must give the **root principal of external account** (or users/roles in the external account) permission to use the CMK.
2. **IAM Policy** must be attached to IAM users/roles in the external account to delegate permissions specified in the Key Policy. This is reliant on the trusted account to ensure that delegated permissions are LEAST PRIVILEGE..

Encryption Context - an additional layer of authentication for KMS API calls

- A optional key-value pair of data that can contain contextual information that you want associated with KMS-protected information. IF encryption context value is used in ENCRYPTION <-> must be used also in DECRYPTION.
- **Additional Authentication Data (AAD)**: encryption context key-value pair is incorporated into AAD in KMS-encrypted ciphertext.
- **Not a secret**: encryption context appears in plaintext in CloudTrail Logs so you can use it to identify/categorise cryptographic operations. Do NOT store sensitive info in the key-value pair.
- **Limiting access/scope**: encryption context can be used to limit access to your resources e.g. only S3 buckets with context `bucket-name:helloworld` can be encrypted/decrypted under the CMK.

Multi-Factor Authentication - can be added via. conditional statement in CMK Key Policy to protect critical KMS calls

- Example: Key Policy with critical KMS calls: `PutKeyPolicy`, `ScheduleKeyDeletion`, `DeleteAlias` and `DeleteImportedKeyMaterial`

```
{
    "Sid": "MFACriticalKMSEvents",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333/user/ExampleUser"
    },
    "Actions": [
        "kms:DeleteAlias",
        "kms:DeleteImportedKeyMaterial",
        "kms:PutKeyPolicy",
        "kms:ScheduleKeyDeletion"
    ],
    "Resource": "*",
    "Condition": {
        "NumericLessThan": {
            "aws: MultiFactorAuthAge": "300"
        }
    }
}
```

Detective Controls

Detective Controls ensures that you properly configure AWS KMS to log the necessary information you need to gain greater visibility into your environment.

CMK Auditing - KMS is integrated natively with CloudTrail

- All KMS calls are automatically logged in files delivered to an S3 bucket that you specify.
- Monitor for specific KMS calls such as `ScheduleKeyDeletion`, `PutKeyPolicy`, `DeleteAlias`, `DisableKey`, `DeleteImportedKeyMaterial` on your KMS keys.
- KMS also produces CloudWatch Events when your CMK is rotated, deleted, and imported key material expires.

CMK Use Validation - validate that your CMKs are being used properly / aligns with best practices

- **AWS Config**: E.g. Config rule `ENCRYPTED_VOLUMES` can be used to validate that attached EBS volumes are encrypted.
- **Key Tags**: A CMK can a tag applied to it, to correlate back to a business category e.g. cost center, application name, owner etc. Use CloudTrail to verify that the CMK being used belongs to the same cost center of the resource that the CMK is used on (e.g. Marketing CMK used on Marketing S3 Storage).

Infrastructure Security

Infrastructure Security provides you with the best practices on how to configure AWS KMS to ensure that you have an agile implementation that can scale with your business, while protecting your sensitive information.

Customer Master Keys (CMKs) are used to:

1. Encrypt DATA BLOCKS of up to **4KB**. OR;
2. Encrypt DATA KEYS, which protect underlying data of ANY SIZE.

CMKs - AWS-managed CMK vs Customer-managed CMKs

- **Creation**: AWS generated || Customer generated
- **Rotation**: Once/3years automatically || Once/year opt-in manually
- **Deletion**: Can't be deleted || Can't be deleted
- **Scope of Use**: Limited to specific AWS service || Controlled via. KMS Key Policy /IAM Policy
- **Key Access Policy**: AWS managed || Customer managed
- **User Access Management**: IAM policy || IAM policy

Customer-managed CMKs - 2 options for creating underlying key material

1. KMS generates cryptographic key material for you.
2. Customer imports their own cryptographic key material. Benefits include:
   - Allows you to meet compliance requirements.
   - Ability to set an expiration time for key material in AWS and manually delete it, but also make it available again in the future.
   - Additional durability and disaster recovery as you can keep the key material outside of AWS.

CMK Key Aliases - abstract CMK users away from underlying Region-specific key ID and key ARN

- **Multi-region apps**: can benefit from using the same key alias to refer to CMKs in multiple-regions without worrying about key ID or key ARN.
- **Recommended alias format**: `alias/<Environment>-<Function>-<Team>` e.g. `alias/development-mfakey-integrations`
- CMK aliases can't be used within policies (Key Policies, IAM Policies, KMS Grants) as mapping of alias to keys can be manipulated outside the policy, which would allow privilege escalation.

Using AWS KMS at Scale - Envelope Encryption

- **Envelope Encryption** can be used to encrypt a unique Data Key, which is used to encrypt plaintext data.
- **Reduce no. of requests to AWS KMS**: decrypt the encrypted Data Key with the CMK once, then cache the plaintext Data Key for repeated use for X period of time.
- **Enable Disaster Recovery** by copying your encrypted data between Regions and only re-encrypting Data Keys with Region-specific CMKs.

Data Protection

Data Protection addresses some of the common use cases for using KMS to protect sensitive info

USE CASE: Encrypting PCI Data Using AWS KMS

- KMS service meets requirements of PCI DSS Level 1 certification.
- **Reduce burden of managing encryption libraries**: encrypt Primary Account Number (PAN) data with a CMK.
- **KMS requests are logged in CloudTrail**: CMK use can be audited easily.

USE CASE: Manage secrets using KMS and S3

1. Create an S3 bucket to hold secrets -> deploy bucket policy to limit access to authorized individuals/services.
2. Each secret is stored using a prefix to allow for granular access control to the secret -> each secret encrypted with a specific customer-managed CMK.
3. Enable S3 access logging or CloudTrail Data Events for audit purposes.
4. A user/service that requires access to the secret assume an identity that has permissions to use both the S3 OBJECT in the bucket and the KMS KEY.

USE CASE: Encrypting Lambda Environment Variables

- Lambda env vars are encrypted using AWS-managed KMS by default.
- **Enable encryption helpers** to individually encrypt env vars using a customer-managed CMK.

USE CASE: Encryption Data within Systems Manager Parameter Store ***Parameter Store Secure String** values are encrypted using AWS-managed or Customer-managed CMKs then decrypted when processing the Secure String value on a managed instance.

USE CASE: Data at Rest Encryption with Amazon S3 (using SSE-KMS)

- **S3 Server-Side Encryption**: Deploy an S3 bucket which enforces all objects are encrypted BEFORE being uploaded to the bucket.
- SSE-KMS Bucket Policy:

```
{
    "Version": "2012-10-17",
    "Id": "PutObjPolicy",
    "Statement": [
        "Sid": "DenyUnencryptedObjectUploads",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:PutObject",
        "Resource": "arn:aws:s3:::YourBucket/*",
        "Condition": {
            "StringNotEquals": {
                "s3:x-amz-server-side-encryption": "aws:kms"
            }
        }
    ]
}
```

USE CASE: Data at Rest Encryption with Amazon EBS

- **Encrypt EBS volumes with KMS**: Data is encrypted at rest + decrypted at the hypervisor of the EC2 that uses the EBS volume on an AS-NEEDED BASIS.
- **Enforce EBS encryption #1**: Upon EBS volume creation requests, check that `CreateVolume` parameter ENCRYPTED=TRUE.
  - If value is not TRUE, prevent IAM entity from creating volume.
- **Enforce EBS encryption #2**: Use CloudTrail to monitor for new EBS volumes being created -> trigger Lambda to respond to unencrypted volume && check KMS key used for encryption (to make sure its the correct one).
  - Lambda can automatically delete EC2 that has the unencrypted volume.
  - Lambda can automatically quarantine EC2 by preventing inbound connections via. Security Group rules.
  - Lambda can simply trigger SNS topic to alert security of unencrypted volume.
  - Lambda can call the `CopyImage` API to create a new encrypted version of the EBS -> automatically attach it to the instance -> delete the old EBS volume.

USE CASE: Data at Rest Encryption with Amazon RDS

- **RDS relies on EBS to provide encryption of database volumes**: Create an encrypted database instance -> RDS creates an encrypted EBS volume to store the database.

- **Enforce EBS encryption via. Lambda**: Lambda to monitor for `CreateDBInstance` calls via. CloudTrail -> ensure `KmsKeyId` parameter is set to the expected CMK -> react to failures.

Incident Response

Use Lambda to automate IR: Monitor CMK actions that indicate compromise -> Lambda disables CMK or other reactive measure.

Deleting and Disabling CMKs

- **MINIMUM 7-DAY to 30-DAYS waiting period is enforced by KMS before a CMK is truly deleted.** `kms:ScheduleKeyDeletion`
- **"Pending Deletion"** state of a CMK means it can't be used for encrypt/decrypt operations.
- **Use CloudTrail to detect kms:ScheduleKeyDeletion** to reverse accidental or malicious key deletion.
- **CMK with Imported Key Material** can be deleted immediately with `kms:DeleteImportedKeyMaterial`, making CMK unusable with NO WAITING PERIOD. If you lose the Key Material locally, then you will NOT be able to access encrypted data anymore.

Security at Scale: Logging in AWS

Control Access to Log Files

The ability to view or modify your log data should be restricted to authorised users.

Configure IAM roles and S3 bucket policies to enforce read-only access to your log files.

Enable AWS MFA on S3 buckets that store CloudTrail logs.

Obtain Alerts on Log File Creation and Misconfiguration

Alerts must be sent for logging creation or misconfiguration.

CloudTrail publishes notifications when logs are CREATED or FAIL/MISCONFIGURED to:

- An S3 bucket: notification is shown via. AWS Console.
- An SNS topic: notification is received via. SMS/email or other AWS services.

Manage Changes to AWS Resources and Log Files

Understanding, preventing changes and unauthorized access to log data is necessary for the integrity of your change management processes and for the ability to comply with internal, industry and regulatory requirements around change management.

CloudTrail logs any AWS API calls made via. AWS Console, AWS CLI and AWS SDKs.

By default, API call log-files are encrypted using SSE-S3 and placed into your S3 bucket.

Modifications to log data can be controlled via. IAM and MFA to enforce read-only access to your S3 bucket storing your CloudTrail log files.

Storage of Log Files

Industry standards and legal requirements may require that log files be stored for varying periods of time.

With CloudTrail, you can:

- Store log-files in S3 for RESILIENCY as S3 is designed for 99.99% durability and 99.99% availability of objects over a given year.
- Aggregate log-files across all regions and multiple accounts to a single S3 bucket.
- Configure your desired EXPIRATION PERIOD on log files written to an S3 bucket.

By default, log-files are stored indefinitely.

Move your log-files to Amazon Glacier to save costs on long-term storage.

Generate Customised Reporting of Log Data

Gaining a CLEAR understanding of activities users have performed and changes made to your IT environment is important.

CloudTrail log-files can be input into industry leading log management and analysis solutions to perform analytics.

CloudTrail produces log-data from a single internal system clock by generating event timestamps in Coordinated Universal Time (UTC) consistent with the ISO 8601 Basic Time and Date format standard.

CloudTrail delivers API calls with detailed info such as

- Type, data and time, location, source/origin, outcome (including exceptions, faults and security-event information), affected resource (data, system etc.) and associated user.
- User identity, time of event, IP address of user, request parameters provided by user, response elements returned by service and optional error code and error message.