**AWS CSA-Pro Notes ACloudGuru Section 4 Security**

**Exam Tips:**

**Multi-Account Management:**

**Know the different models and best practices for cross-account management of security**

* Identity Account Management:
  + Manage all user account in one location. Use trust relationships from IAM roles in Sub accounts to identity account to grant temp access.
  + Different accounts can be by business unit, deployment environment, or geography
* Centralized Logging Repository:
  + Have one account for logs and only logs. You can make this account immutable with very limited access.
  + You can use Service Control Policies to prevent sub-accounts from changing logging settings
* Publishing Account Structure:
  + Common repository for AMIs, Containers, Code.
  + Permits sub-accounts to use pre-approved standardized services or assets
* Information Security Account:
  + Hybrid of consolidated security and logging
  + Allows one point of control and audit
  + Logs cannot be tampered with by sub account users
* Central IT Account:
  + It can manage IAM users and groups while assigning to sub-acc roles
  + It can provide shared services and standardized assets that adhere to corporate policy

**Know how roles and trusts are used to create cross-account relationships and authorizations**

* Not talked about in videos, but basically
  + You create a role and you define what that role allows anyone assuming it to do via the IAM policy.
  + Then you create a trust policy which basically tells the role who do I trust to assume this role

**Network Controls and Security Groups:**

**Know the differences and capabilities of NACLs and SGs**

* SG:
  + Stateful, only can allow traffic, at the instance level
* NACL:
  + Stateless, can allow and deny traffic, at the subnet level

**NACLs are stateless**

**Get some hands-on with NACLs and SGs to reinforce your knowledge**

**Remember the ephemerals**

* Only really required for outbound port of NACLs

**AWS Directory Services:**

**Understand the types of Directory Services offer by AWS-especially AD Connector and Simple AD**

* AWS Cloud Directory: Cloud-native directory to share and control access to hierarchical data between apps
* Cognito: Sign-up and sign-in functionality that scales to millions of users and federated to public social media services
* Directory service for Microsoft AD: AWS-managed full Microsoft AD on windows server 2012 R2
* AD Connector: Allows on-prem users to log onto AWS services with their existing ad creds. Allows EC2 instances to join AD domain
* Simple AD: low-scale, low cost, ad implementation based on samba

**Understand use cases for each type of Directory Service**

* AWS Cloud Directory: Best for cloud apps that need hierarchical data with complex relationships
* Cognito: Develop consumer apps or SaaS
* AWS Directory Service for Microsoft AD: Enterprise that want hosted Microsoft AD or you need LDAP for Linux apps
* AD Connector: SSO for on-prem employees and for adding EC2 instances to the domain
* Simple AD: Simple user directory, or you need LDAP compatibility

**Be familiar with how on-prem Active Directory implementation might connect to AWS and what functions that might enable**

**Credentials and Access Management:**

**Know IAM and its components**

* Users, Groups, Roles, Policies

**Know how to read and write IAM policies in JSON**

**Understand Identity Brokers, Federation, and SSO**

**Know options and steps for temp auth**

**Encryption:**

**Know differences between AWS KMS and CloudHSM and use cases**

* KMS:
  + Multi-Tenant, Cost less, Integrates with AWS services better, FIPS 140-2 level 2 compliance, AWS managed
* CloudHSM:
  + Single-Tenant, Cost more, Integrates with 3rd party things better, FIPS 140-2 level 3 compliance

**The test will likely be restricted to the “classic” CloudHSM**

**Understand AWS Certificate Manager and how it integrates with other AWS services**

* Integrates with CloudFront, API Gateway, and App Load Balancers to get easy encryption in transit.

**DDoS Attacks:**

**Understand what they are and some best practices to limit your exposure**

**Know some options to mitigate them using AWS services**

**IDS/IPS:**

**Understand the difference between IDS and IPS**

* IDS: Detection server to proactively detect intruders
* IPS: Protection server used to proactively respond to intruder threats. Like missed passwords

**Know what AWS services can help with each**

* CloudWatch Logs

**Understand the differences between CloudWatch and CloudTrail**

* CloudWatch: AWS Monitoring and Logging tool for usually Instance level logging or service level logging
* CloudTrail: Logs API calls made to your AWS console. Can see what user turned off an instance at what time

**Service Catalog:**

**Know that it allows users to deploy assets through inheriting rights**

**Understand how Service Catalog can work in a multi-account scenario**

**Concepts:**

**Shared Responsibility Model:**

* Security of the Cloud: AWS does this and it has to do with the security of the actual on-site data centers
  + Software
  + Compute, Storage, DB, Networking
  + Hardware
  + Regions, AZ’s, Edge Locations
* Security in the Cloud: you do this and it has to do with how your architect your applications with a security mindset
  + Customer Data
  + Platform, Apps, IAM
  + OS, Network and Firewall configs
  + Client-Side data, Encryptions and Date Integrity
  + Server-Side encryptions
  + Networking Traffic Protection

**Principle of Least Privilege:**

* Give users (or services) nothing more than those privileges required to perform their intended function
* And only when they need it \*

**Security Facets:**

**Identity:**

* Who are you?
* Root Account User, IAM User, Temp Security Credentials

**Authentication:**

* Prove that you are who you say
* MFA Authentication, Client-side SSL certs

**Authorization:**

* Are you allowed to do this?
* IAM policies

**Trust:**

* Do other entities that I trust also trust you? Cross Account Access, Federation

**Typical Components of Security:**

**Identities:**

* People, Objects, Other computers

**Identity Provider:**

* Cognito
* Facebook, Google, Twitter
* AD

**Identity Store:**

* Inside the Identity Provider
* Stores the Identities
* Additional meta data about those Identities

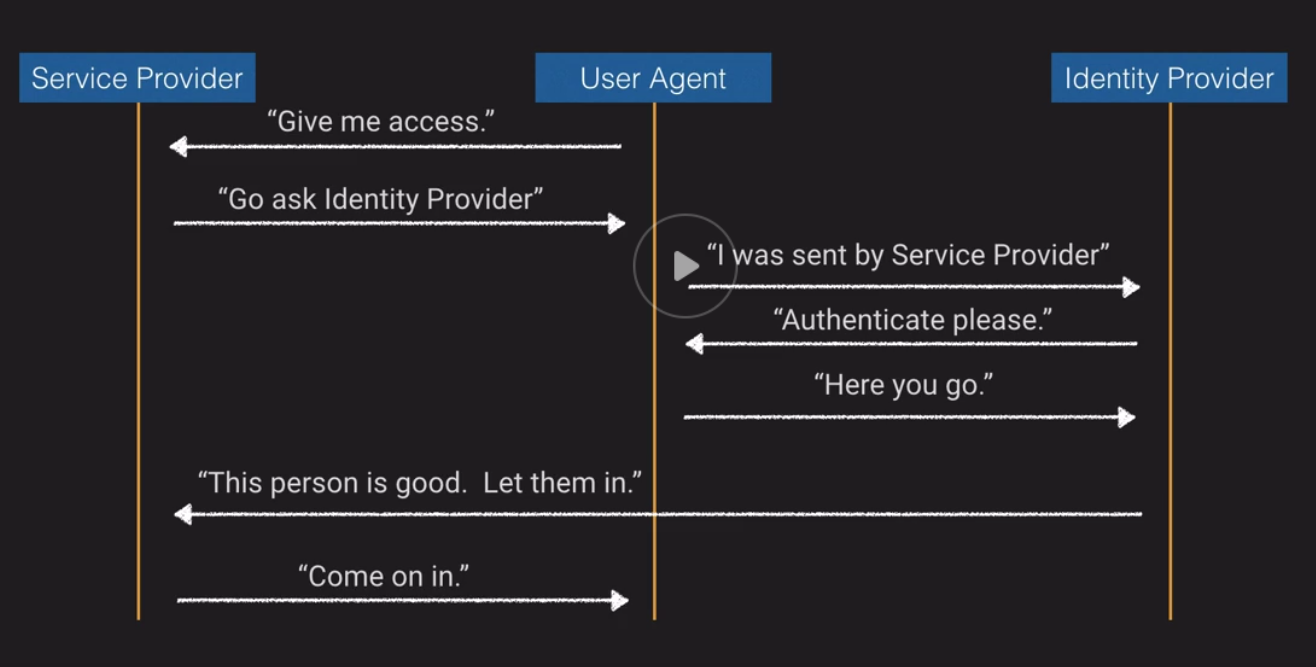
**Identity Broker:**

* Takes requests from the Identities and run those up against the Identity Store

**Federation:**

* When the Identity Broker reaches out to other Identity Providers such as Facebook, Google, Cognito

**Typical Authentication Flow:**

****

**SAML vs. OAuth vs. OpenID:**

**SAML 2.0:**

* Can handle both authorization and authentication
* XML-based protocol
* Can contain user, group membership and other useful info
* Assertions in the XML for authentication, attributes, or authorization
* Best suited for Single Sign-on for enterprise users

**OAuth 2.0:**

* Allow sharing of protected assets without having to send login credentials
* Handles authorization only
* Issues token to client
* Application validates token with Authorization Server
* Delegate access, allowing the client applications to access information on behalf of user
* Best suited for API authorization between apps

**OpenID Connect:**

* Identity layer build on top of OAuth 2.0, adding authentication
* Uses REST/JSON message flows
* Supports web clients, mobile clients, Javascript clients
* Extensible
* Best suited for SSO for consumer

**AWS Artifact-** A central place to find supporting documentation of various compliance reports using this service.

**Multi-Account Management:**

* Most large orgs will have multiple AWS accounts
* Allows for segregation of duties, cost allocation, and increased agility
* Need methods to properly manage and maintain them

**When should you use Multiple Accounts?**

* Do you require administrative isolation between workloads?
* Do you require limited visibility and discoverability of workloads?
* Do you require isolation to minimize “blast radius”?
* Do you require strong isolation of recovery and/or auditing data?

**AWS tools for Account Management:**

**AWS Organizations:**

* Manage multiple accounts in a hierarchical fashion straight from the AWS console

**Service Control Policies:**

* Policies that you can apply to sub accounts for strict rules of lower accounts

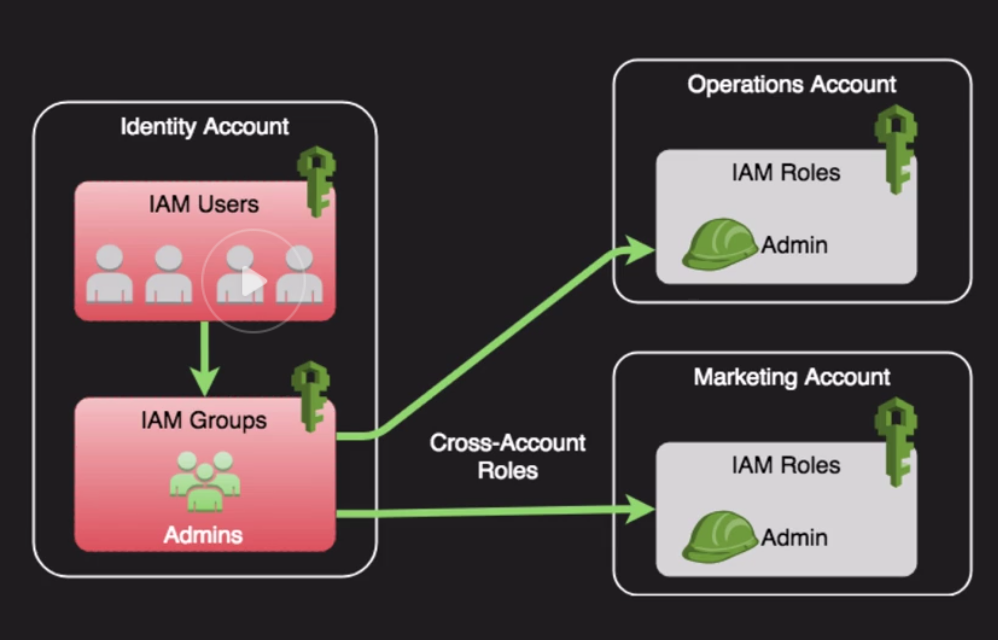
**Tagging**

**Resource Groups**

**Consolidated Billing**

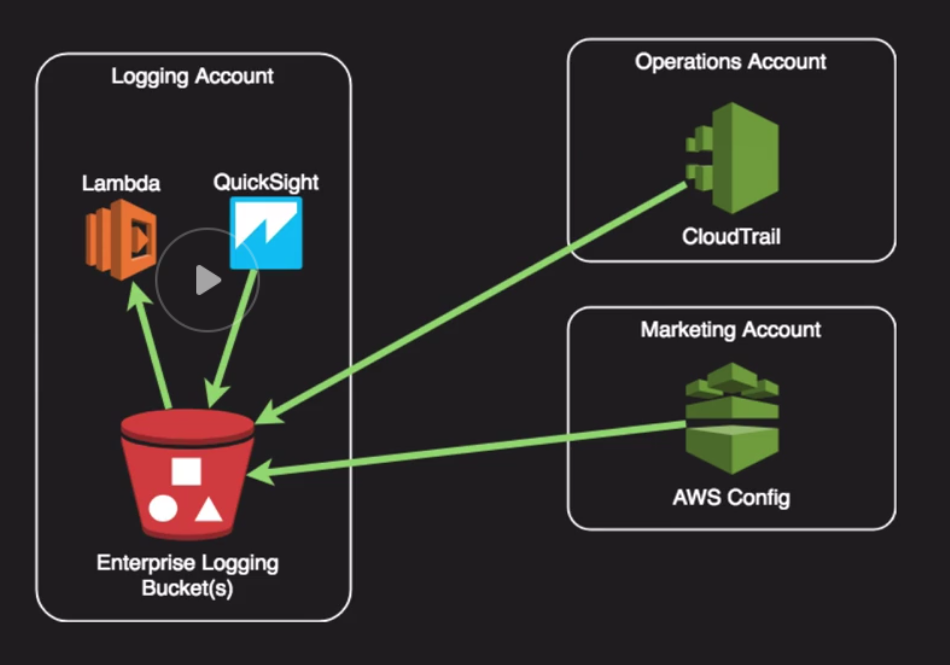
**Account Structures:**

**Identity Account Structure:**

* Manage all user accounts in one location
* Users trust relationships from IAM roles in sub-accounts to Identity Accounts to grant temp access
* Variations include by Business Unit, Deployment Environment, Geography
* **Trust Relationships:** IAM roles use these to figure out what type of Identity is trusted to assume a role. Could be an EC2 instance is trusted or an IAM user etc…

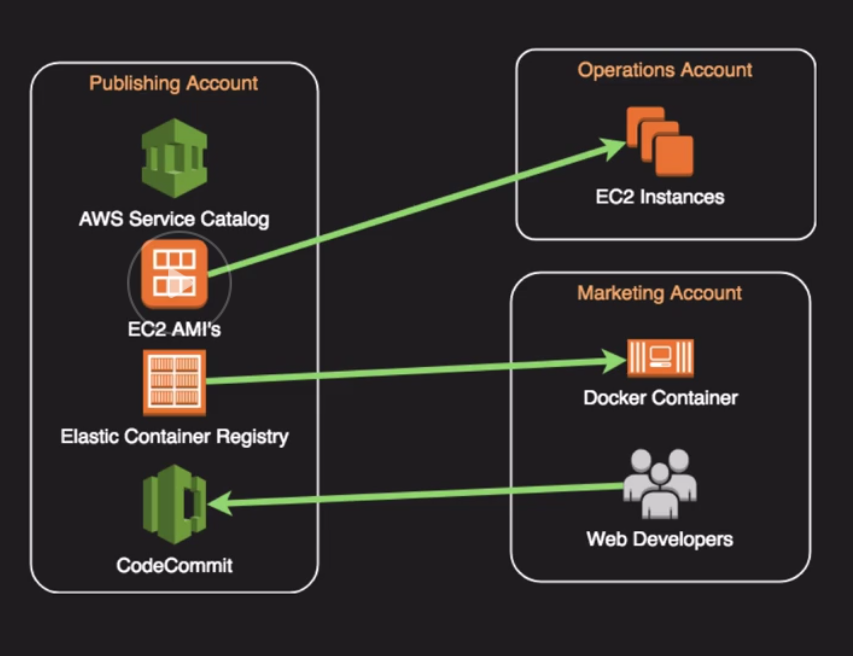
**Logging Account Structure:**

* Centralized Logging Repository
* Can be secured so as to be immutable
* Can use Service Control Policies to prevent sub-accounts from changing logging settings

****

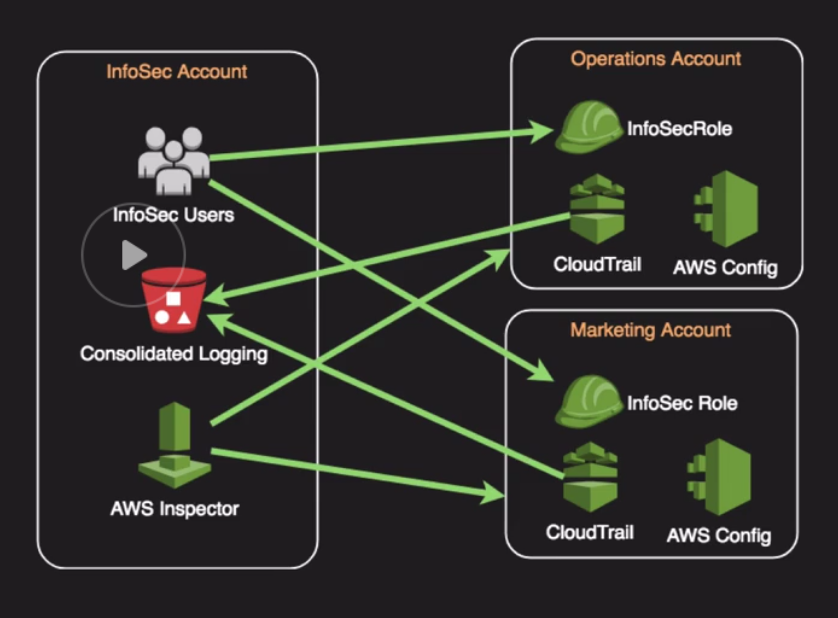
**Publishing Account Structure:**

* Common repository for AMI’s, Containers, Code
* Permits sub-accounts to use pre-approved standardized services or assets

****

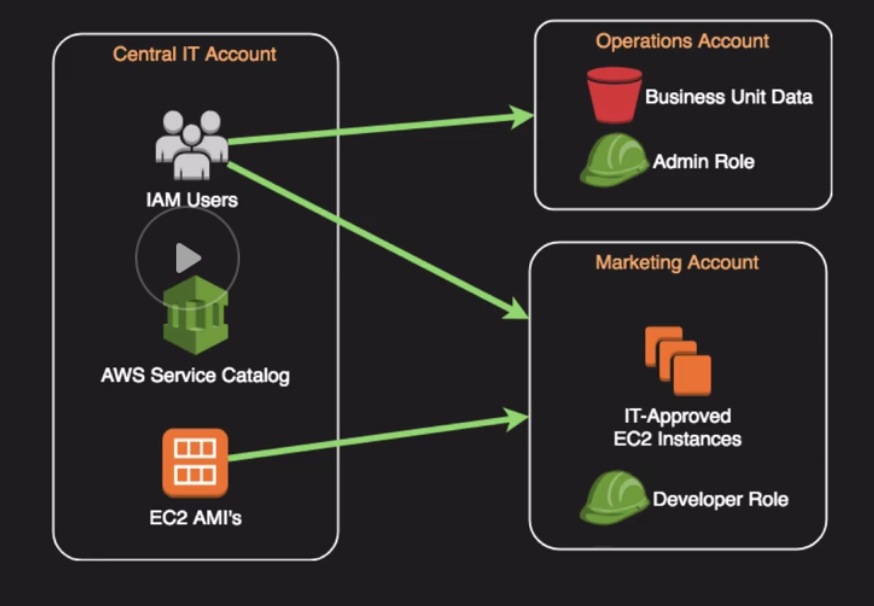
**Information Security Account Structure:**

* Hybrid of consolidated security and logging
* Allows one point of control and audit
* Logs cannot be tampered with by sub-account users

****

**Central IT Account Structure:**

* IT can manage IAM users and groups while assigning to sub-account roles
* It can provide shared services and standardized assets that adhere to corporate policy

****

**Network Controls and Security Groups:**

**Security Groups:**

* Virtual firewalls for individual assets (EC2, RDS, AWS Workspaces)
* Controls inbound and outbound traffic for TCP, UDP, ICMP, or custom protocols
* Port or port ranges
* Inbound rules are by Source IP, Subnets, or other Security group
* Outbound rules are by Destination IP, Subnet, or other Security groups
* Remember ephemeral ports for Outbound if you need them. (Not necessary for Security groups because they are Stateful)

**Network Access Control Lists:**

* Additional layer of security for VPC that acts as a firewall
* Apply to entire subnets rather than individual subnets
* Default NACL allows all inbound and outbound traffic
* NACLs are stateless – meaning outbound traffic simply obeys outbound rules – no connection is maintained
* Can duplicate or further restrict access along with Security Groups
* Ephemeral Ports are important for NACLs on the outbound

**Why use SG’s and NACLs?**

* NACLs provide a backup method of security if you accidentally change your SG to be too permissive
* Covers the entire subnet so users who create new instances and fail to assign a proper SG are still protected
* Part of a multi-layer Least Privilege concept to explicitly allow and deny

**AWS Directory Services:**

**Types of Directory Services Offered:**

**AWS Cloud Directory:**

* Cloud-native directory to share and control access to hierarchical data between applications
* Best for Cloud apps that need hierarchical data with complex relationships

**Amazon Cognito:**

* Sign-up and sign-in functionality that scales to millions of users and federated to public social media services
* Develop consumer apps or SaaS

**AWS Directory Service for Microsoft Active Directory:**

* AWS-managed full Microsoft AD (standard or enterprise) running on Windows Server 2012 R2
* Best for enterprises that want hosted Microsoft AD or you need LDAP for Linux apps

**AD Connector:**

* Allows on-prem users to log into AWS services with their existing AD credentials. Also allows EC2 instances to join AD domain
* Best for SSO for on-prem employees and for adding EC2 instances to the domain

**Simple AD:**

* Low scale, low cost AD implementation based on Samba
* Best for simple user directory, or you need LDAP compatibility

**AD Connection vs. Simple AD:**

**AD Connector:**

* Must have existing AD
* Existing AD users can access Aws assets via IAM roles
* Supports MFA via existing RADIUS-based MFA infrastructure

**Simple AD:**

* Stand-alone AD based on Samba
* Supports user accounts, groups, group policies, and domains
* Kerberos-based SSO
* MFA not supported
* No Trust Relationships

**Credential and Access Management:**

**AWS Security Token Service:**

* Grant temp access via AD or web federation

**Amazon Cognito:**

* Used more in mobile app development for storing mobile app users at large scales

**AWS Secrets Manager:**

* Store passwords, encryption keys, API keys, SSH keys, PGP keys
* Alternative to storing passwords or keys in a “vault”
* Can access secrets via API with fine-grained access control provided by IAM
* Automatically rotate RDS database creds for MySQL, PostgreSQL, and Aurora
* Better than hard-coding creds in scripts or apps

**Encryption:**

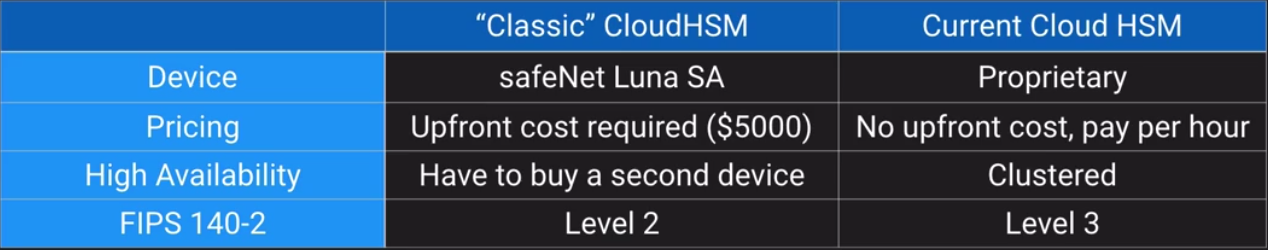
**2 State of Encryption:**

* Encryption at REST:
  + Data is encryption where it is stored such as on EBS, on S3, in an RDS database, or in an SQS queue waiting to be processed.
* Encryption in Transit:
  + Data is encrypted as it flows through a network or process, such as SSL/TLS for HTTPS, or with IPSec for VPN connections.

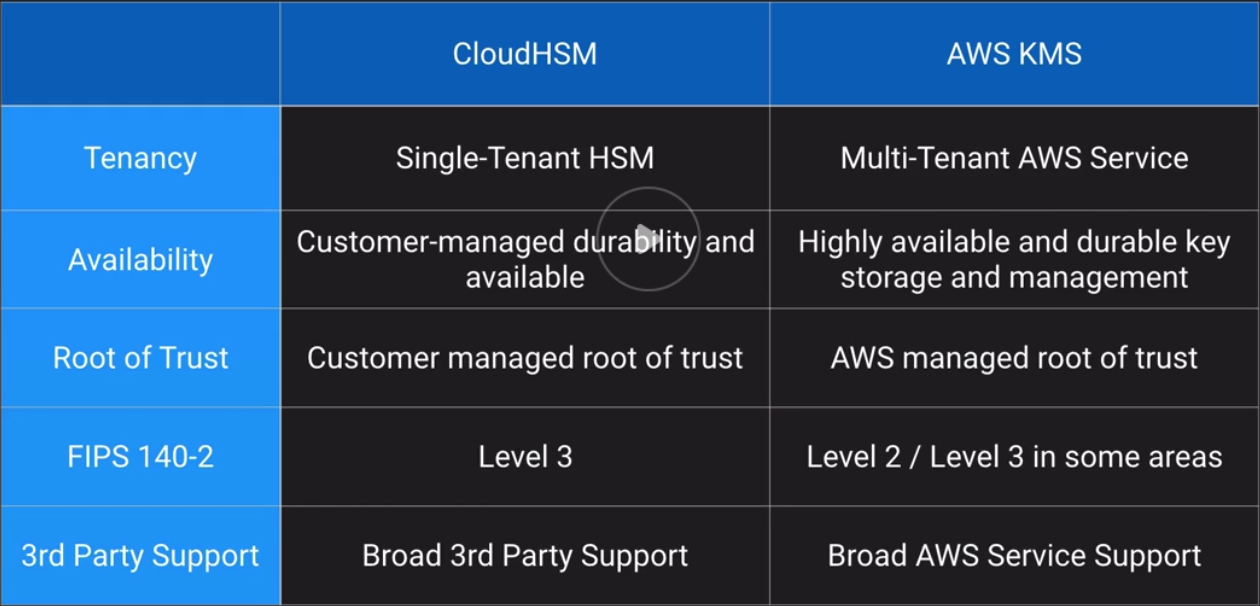
**KMS (Key Management Service):**

* Key storage, management, and auditing
* Tightly integrated into many AWS services like Lambda, S3, EBS, EFS, DynamoDB, SQS, etc..
* You can import your own keys or have KMS generate them
* Control who manages and accesses keys via IAM users and roles
* Audit use of keys via CloudTrail
* Differs from Secret Manager as its purpose-build for encryption key management
* Validated by many compliance schemes

**CloudHSM:**

* Dedicated hardware device, Single Tenanted
* Must be within a VPC and can access via VPC Peering
* Does not natively integrate with many AWS services like KMS, but rather requires custom application scripting
* Offload SSL from web servers, act as an issuing Cam enable TDE for Oracle DBs

**CloudHSM vs KMS:**

****

**AWS Certificate Manager:**

* Managed service that lets you provision, manage, and deploy public of private SSL/TLS certs
* Directly integrated into many AWS services like CloudFront, ELB, and API Gateway
* Free public certs to use with AWS services
* Supports wildcard domains to cover all your subdomains
* Managed certificate renewal
* Can create a managed Private Certificate Authority as well for internal proprietary apps, services, or devices

**Distributed Denial of Service Attack:**

**DDOS:**

**Minimize attack surface:** Use NACLs, SGs, and well-designed VPCs

**Scale to absorb attack:** Auto-Scaling Groups, AWS CloudFront, Static Web Content via S3

**Safeguard Exposed Resources:** Route 53, AWS WAF, AWS Shield

**Learn normal behavior:** AWS GuardDuty, CloudWatch

**Have a plan:** Relies on you

**IDS/IPS:**

**Intruder Prevention and Detection:**

**Intruder Detection System:** Watches the network and systems for suspicious activity that might indicate someone trying to compromise a system

**Intruder Prevention System:** Tries to prevent exploits by sitting behind firewalls and scanning and analyzing suspicious content for threats

A big part of these systems is a log collection tool used with them. CloudWatch Logs or 3rd party tools such as Splunk.

**CloudWatch vs CloudTrail:**

**CloudWatch:**

* Log events across AWS services
* Higher level comprehensive Monitoring and Eventing
* Log from multiple accounts
* Logs stored indefinitely
* Alarms history for 14 days

**CloudTrail:**

* Log API activity across AWS services
* More low-level granular
* Log from multiple accounts
* Logs stored to S3 or CloudWatch indefinitely
* No native alarming; Can use CloudWatch alarms

**AWS Service Catalog:**

* Framework allowing admins to create pre-defined products and landscapes for their users
* Granular control over which users have access to which offerings
* Makes use of adopted IAM roles so users don’t need underlying service access
* Allows end users to be self-sufficient while upholding enterprise standards for deployments
* Based on CloudFormation templates
* Admins can version and remove products. Existing running product versions will not be shut down

**AWS Service Catalog Constraints:**

**Types:**

**Launch Constraint:**

What: IAM role that Service Catalog assumes when an end-user launches a product

Why: Without a launch constraint, the end-user must have all permissions needed within their own IAM creds

**Notification Constraint:**

What: Specifies the Amazon SNS topic to receive notifications about stack events

Why: Can get notifications when products are launched or have problems

**Template Constraint:**

What: One or more rules that narrow allowable values an end-user can select

Why: Adjust product attributes based on choices a user makes